PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-153640
(43)Date of publication of application : 06.06.2000

(51)Int.CI. B41J 2/52
H04N 1/23

(21)Application number : 11-327652
(22)Date of filing : 18.11.1999

(71)Applicant : XEROX CORP
(72)Inventor : CURRY DOUGLAS N

(30)Priority
Priority number : 98 195165 Priority date : 18.11.1998 Priority country : US

(54) PRINT METHOD AND PRINTER CONTROLLER

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a printer controller and a method therefor which use print data and printer control commands so that a laser power to be used for forming objects of each type to page images defined with use of a PDL or the like is controlled or selected to be optimum.
SOLUTION: The print method includes a step of distinguishing a byte space of image data including data of different types, a step of printing byte image data of a first type with use of a laser power drive signal at a first step point, and a step of printing byte image data of a second type with use of the laser power drive signal at a second step point. The first step point is different from the second step point.

LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] The print method that have a step which distinguishes between cutting tools of image data containing data of a different type, a step which prints cutting tool image data of the 1st type on the 1st step point using a laser power driving signal, and the step which prints image data of a cutting tool of the 2nd type on the 2nd step point using said laser power driving signal, and said 1st step point differs from said 2nd step point.
[Claim 2] It is the printer controller which adjusts a driving signal set point used in order to print the cutting tool's image data based on a type of an object corresponding to a cutting tool's image data. It has an image printing terminal which receives print data and meta-bit data. It has an image processing system containing an object optimization image formation modulation subsystem with which said image printing terminal is selectively controlled by meta-bit data transmitted to said image printing terminal. Said meta-bit controls said object optimization image formation modulation subsystem to choose between driving signal set points which change with types of an object with which said cutting tool's image corresponds. A printer controller which affects said formation of said object corresponding to image data said whose driving signal set point is said cutting tool.
[Claim 3] A printer controller according to claim 2 which is built into an image formation device and chosen from a group to whom said image formation device changes from a facsimile machine, a printer, a digital process copying machine, and a raster output scanner.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] This invention relates to the method and equipment for adjusting max, saturation laser, or LED power in a printing equipment. This invention relates to the method for controlling exposure level at details based on the type of an image object by which current exposure is carried out more.
[0002]
[Description of the Prior Art] The creator (implementer) of the page image for which it had been depending on the GURAFIKU artist for some time in order to compound and print a page image with development of a digital color workstation, a copying machine, and a printer can use instead the computer connected to the digital color copying machine / printer, and can generate, compound and print a page image by itself.
[0003] However, in order to generate a page image and to disassemble a page image into a print engine instruction, the digital system of such conventional technology for controlling a print engine was single, and was processing the page image as an image of one. Thus, in the page image optimized for a text / line art, when a RF halftone screen is used, the text portion of the page image is very sharp. However, the fixed color portion of a page image includes clear spots-ization from a printer noise. Furthermore, the color portion by which the page image was sampled, and a sweep (sweep) portion include clear outline formation by lack of sufficient gray level which may be used on a RF screen.
[0004] In the page image optimized for the big fixed color portion, the halftone screen designed especially in order to hide the instability of a printer generates the fixed color field which is quality, does not have a text and does not have the artifact. However, the sharpness of a text falls, the gray value for every tint is seldom related, and the color portion and sweep portion which were sampled cannot be permitted. Since each dot level is independently designed not related with other level, the step of the gray level is not smoothly carried out from one level to the following level.
[0005] In the page image optimized to the color portion and sweep portion which were sampled, since a low frequency halftone screen is used by the gray level which can be used, a sweep portion and the sampled color portion show higher quality. However, a text is expressed with low quality and a fixed color portion shows clear pattern-ization.
[0006] However, in the conventional system which processed each image as a single bitmapped image or a cutting tool map image, optimizing an image to all one types of object needed to compromise with the image quality of the object of other types. therefore , the digital color copying machine / printer , and the method of maintain the advantage which can be use when generate a page image using a microcomputer , and effectiveness be required of the conventional technology at the same time it output to the print engine which generate a page image , decompose and can optimize the print property of the object according to individual similarly in the synthetic GURAFIKU art field by handicraft .
[0007] GrAphicAl which is used with a Page Description Language [like PostScriptTM] (PDL) such whose a page image is, InterpressTM, and WindowsTM DisplAy InterfAces (GDI) and

Hewlett-PAckArd Printer CommAnd LAnguAge (PCL-5) etc. -- it is used and generated.
[0008] It is known that the laser beam printer product marketed will expose "monochrome" image using the powerful laser power which is not common. Enhancement of the contrast between a white image and a black image is benefited for this, and the enclosure field by this fault exposure is liked aesthetic for those who see. All the image ranges are provided with this enhancement laser power.
[0009] However, this reinforced contrast makes it impossible to generate a thin line in connection with it. Furthermore, although fault exposure of a text and/or a line art is liked aesthetic, since the reinforced laser power carries out fault exposure of the halftone cell, fault exposure of a halftone image is disadvantageous. Carrying out fault exposure of the halftone cell decreases the control covering the whole dynamic range. Since enhancement of laser power covers all the dynamic ranges of a print engine and influences transition of a form function, this is produced. If laser power is increased especially, a form function is saturated too much quickly and cannot change in the high edge of a dynamic range.
[0010]
[Problem(s) to be Solved by the Invention] This invention makes it a technical problem to offer the printer controller equipment and the method of using print data and printer control command so that the laser power used in order to form the object of each type in the page image defined using PDL etc. may be controlled or chosen the optimal. Print data and printer control command may be changed from the page image defined using PDL, and may be offered according to other well-known mechanisms in the conventional technology.
[0011] This invention generates the "meta-bit" information which includes laser power information further based on the object type of the various objects which form a page image, i.e., the information about the method of improving each cutting tool's image data most a rendering, and offers the object optimization printer control unit and method of sending these meta-bit data to an image printing terminal (IOT).
[0012] This invention offers the object optimization printer control unit and method of determining laser power automatically further based on the object type to the object of each independence of a page image.
[0013]
[Means for Solving the Problem] A page image which is described as an example of a step of equipment used and a method with this invention using PDL, and is stored as a series of PDL commands is inputted into an object optimization electro nick subsystem (OOESS). A PDL decomposition means disassembles a page image described by PDL to a data structure showing a page image. In this data structure, information relevant to an independent image object is held. This information includes information relevant to a property according to individual of others, such as a type of an object and the maximum laser modulation set point, a color, the optimal color space, and layer (layer) information.
[0014] Once it changes a page image with which a PDL decomposition means was described by PDL, an image printing terminal (IOT) and a command instruction generating means will change a data structure into a series of command instructions, color assignment, and a meta-bit rendering instruction for every scan line of a page image at a copying machine/printer, and a general twist target. In order to determine a type of each object, you may be automatically generated by meta-bit generating means to analyze each object, and a meta-bit may be clearly set up by page image creator between generation of PDL description of a page image. If a command instruction, color assignment, and a meta-bit are generated for every scan line of a page image, they will be outputted to an IOT controller.
[0015] An IOT controller receives a command instruction, color assignment, and a meta-bit which were generated between decomposition processings. An IOT controller combines a fixed color and sampled color data, and transmits it to IOT with meta-bit data.
[0016] In a gestalt of implementation of instantiation of the 1st of IOT, IOT can include two or more halftone screen generators, a threshold circuit, a color space conversion circuit, and a laser power section circuit. Cutting tool width-of-face color data and a meta-bit which are outputted from an IOT controller are inputted into IOT. A meta-bit is used in order to determine

a threshold circuit [which halftone generator or] to be used to color data, and which color conversion is applied by color space conversion circuit. Moreover, a meta-bit is used in order to choose an alternative laser power set point which should be used in order to print image data, i.e., the maximum laser reinforcement. If IOT generates raster data from color data and a meta-bit, raster data containing laser data on the strength will once be outputted to a marking subsystem, in order to form a page image on an output sheet.

[0017] Or sample color-data compression / compression discharge circuit, mask-data compression / compression discharge circuit, fixed color compression / compression discharge circuit, and a command data compression / compression discharge circuit may be included in an IOT controller of a bus interface and integrated KOMBANA (coupler). Therefore, as for these compression/compression discharge circuits, data of various types of a data structure is compressible the optimal for a transfer to an IOT controller on a system bus. Furthermore, in a gestalt of implementation of instantiation of the 1st of an IOT controller, a color space converter is included in an IOT controller. By including a color space converter in an IOT controller, color space conversion to an object from which a page image differs can be exactly performed before a print, and may be further optimized based on an object type. Furthermore, including a color space converter in an IOT controller extends the range of IOT which may be used with a system of this invention. Or other gestalten over these compression/compression discharge circuits may be used.

[0018] Furthermore, since color data outputted to IOT are generated, two color space converters may be used. Thus, 4 bytes of data showing each of four color layer separation, and C, Y, M and K may be outputted simultaneously. By outputting simultaneously color data to all 4 color layer separation, a high page output is attained [rather than] to IOT, and the larger range of IOT may be used with a system of this invention.

[0019] A step which distinguishes between cutting tools of image data which contains data of a type with which the 1st modes differ as a concrete mode of this invention, A step which prints cutting tool image data of the 1st type on the 1st step point using a laser power driving signal, It is the print method that have a step which prints image data of a cutting tool of the 2nd type on the 2nd step point using said laser power driving signal, and said 1st step point differs from said 2nd step point.

[0020] It is the printer controller which adjusts a driving signal set point used in order that the 2nd mode may print the cutting tool's image data based on a type of an object corresponding to a cutting tool's image data. It has an image printing terminal which receives print data and meta-bit data. It has an image processing system containing an object optimization image formation modulation subsystem with which said image printing terminal is selectively controlled by meta-bit data transmitted to said image printing terminal. Said meta-bit controls said object optimization image formation modulation subsystem to choose between driving signal set points which change with types of an object with which said cutting tool's image corresponds. Said driving signal set point is the printer controller which affects said formation of said object corresponding to image data which is said cutting tool.

[0021] In the 2nd mode, a printer controller is built into an image formation device, and the 3rd mode is chosen from a group to whom said image formation device changes from a facsimile machine, a printer, a digital process copying machine, and a raster output scanner.

[0022]

[Embodiment of the Invention] As mentioned above, unless a creator (generation person) covers the whole page image and inserts a printer dependence halftone clearly, the conventional processing for disassembling the page image described by the Page Description Language (PDL) needs to reach a compromise, when printing the bit map or cutting tool map generated by disassembling the page image described using PDL. The following description may be used with the object optimization rendering system and the method that all above-mentioned systems equivalent to PDL are described below, although referred to only to PDL.

[0023] In the conventional system for disassembling and printing the page image described using PDL as mentioned above, namely, the various objects which consist of a page image It is changed into a device coordinate from a PDL coordinate, and a stroke is thickly made thick. An

object is changed into a series of boxes corresponding to the location of the object on a scan line, respectively, and these boxes are loaded to a bit map, when it is a cutting tool map (or an image black/white). By writing the box formed by decomposing the object of a page image in a cutting tool map, all distinction during the type with which objects differ is lost.
[0024] With this, the page image reversely described by PDL in this invention is disassembled so that distinction between the object types which consist of a page image may be maintained. By maintaining the object type of the object from which it differs in a page image, processing of a different object type can be optimized to the type of an object.
[0025] Before, in order to form the text image of a sharp edge at the same time coarse screen halftone generating processing is used, in order to form the background of concentration of changing continuously, detailed halftone generating processing was used. The optical density which changes continuously [a background] may be maintained by using optimal halftone generating processing for every type of an object. Simultaneously, the optimal halftone generating processing for maintaining the sharp edge of a text image may be maintained.
[0026] Thus, while being used in order to form the text image with which the RF halftone screen was optimized, it may be used in order that a hue (tint) assignment halftone set may form the block with which the fixed color was optimized, and it may be used in order that a low frequency halftone screen may form further the color picture and sweep (sweep) which were optimized and sampled.
[0027] By maintaining the object type of the various objects which consist of a page image, other features of an object may fully be optimized like color space conversion of the object to the cyanogen (C) used by IOT from the color space used by the Page Description Language, a Magenta (M), yellow (Y), and (Black B) color space. Each step of color transform processing may actually be optimized like undershirt color clearance processing and tone (gradation) playback curvilinear processing. Similarly, compression processing can be optimized by the object and, thereby, makes min the transition load over the storage resource (resource) and the disassembled page image consumed.
[0028] Furthermore, depending on the degree of the optimization demanded, the level from which distinction of an object type differs may be offered. That is, probably, in some instances (example), it will be enough to direct that an object is colored in a fixed color and to optimize a halftone screen and all processing functions like color space conversion on this level. In other instances, in order to specify one of the halftone screen frequency of many of color space many of [possible / possible one or], or the angles, the level of additional distinction may be required and those all are suitable for a fixed color object. [many of]
[0029] Furthermore, a processing function is optimized in order to offer the suitable level of the laser beam exposure used in order to form each object in an image. In order to form the outstanding halftone dot, a form function and the whole dynamic range of the optical exposure to a shadow (shading) must continue and cohere through a mid tone from highlights. In halftone formation, the problem which originates in fault exposure of a halftone dot like the case where fault exposure of a text and/or the line art is carried out will not be left with an unexposed field not exposed in the range, if a form function moves to the mid tone portion of a dynamic range. Therefore, the halftone acquired serves as perfect overlap, and a form function goes into a shadow field, when [of all dynamic ranges] it is still on the way. Therefore, since the form function has already changed to the place of a dynamic range which is in the high end of a dynamic range from a certain place low-end, the high end of all dynamic ranges does not influence the print quality of an image.
[0030] However, the appearance of a text and a line art is more desirable when fault exposure of a text and/or the line art field is carried out. A reason is that contrast will be emphasized if a text and/or a line art field are exposed by higher power. Although fault exposure looks better, the precision in the rendering of a very thin line will be lost in all images. However, the lost precision is not large as compared with the appearance by which the text and/or the line art were improved. Therefore, it is advantageous to make it larger than the laser power between processings of the laser power between processings of a text and/or a line art of a halftone dot.
[0031] These differences between a text and line art generation, and halftone generation are

based on size and the problem of control. The size of the feature according to individual of the object in a text or a line art object is larger than the feature according to individual in a halftone object. Therefore, forming a text or a line art generates an image coarser than the image generated by half toning. if fault exposure of a text or the line art is carried out, since it will be estranged without the feature according to individual seldom approaching, as for fault exposure, the time of fault exposure of the halftone dot being carried out and ** do not fall a form function to whenever. Furthermore, forming a text and a line art does not need control of a tone as well as halftone formation. In half toning, a form function changes from a circle configuration to a diamond configuration, and it returns to a circle configuration as the image concentration of a halftone dot changes from highlights concentration to shadow concentration through mid tone concentration along with all the dynamic ranges of a print engine. In a full color image, since exact control of the field covered by these form functions controls the color perceived by those whom the ratio of the field covered by a Magenta, yellow, and cyanogen looks at, it is important.
[0032] The object optimization electronic subsystem (OOESS) 100 for changing into the raster data which can be used with the image printing terminal (IOT) 170 to the PDL form of image data, as shown in drawing 1 is Sun of a desirable general purpose computer like a personal computer, California, and Maung Teng. Miycrosystems It is provided by engineering workstation like SunSpArcTM by the shrine manufactured, a minicomputer, etc. OOESS100 can include an internal PDL file source means like the program for generating the PDL expression of a page image like a program which generates PostScriptTM and an InterPressTM compatible document, the program which generates the GDI expression of a page image, and the program which generates the graphical commands set expression of a ** page image. One type of a graphical commands set is Hewlett-PAckArd used in order to operate a laser beam printer and/or an ink jet printer. It is the commands set of PCL-5. Therefore, it should be understood that term"PDL" should be interpreted as including all the types that describe a page image of expression instead of generating the bit / cutting tool map of a page image.
[0033] Or it can be generated, and the PDL expression of a page image can be rather received from a certain remote PDL file source means 112 like the general purpose computer of a remote place connected to OOESS100 through nonvolatile memory, a local network, or a modem rather than it is decomposed directly. Therefore, it should be understood that the PDL file showing a page image can obtain from all the conventional sources.
[0034] Once the PDL file showing a page image is inputted into OOESS100, it will be transmitted to memory 150 through a bus 114. Next, a PDL file is decomposed by the PDL decomposition means 130. The PDL decomposition means 130 reads the PDL file stored in memory 150, and forms the data structure which decomposes it and is shown in drawing 28 thru/or drawing 31 . The data structure shown in drawing 29 contains the rendering tag which directs an object list and an object type. Actuation of a PDL decomposition means is stated to details below.
[0035] If the PDL decomposition means 130 generates a data structure in the memory 150 which stores the various objects of the page image generated from a PDL expression, reading appearance of the data structure stored in memory 150 will be carried out by the command instruction generating means 140. The command instruction generating means 140 changes into a series of command instructions corresponding to color data, bit mask data, and meta-bit data the data structure stored in memory 150 for every scan line. These command instructions, color data, bit mask data, and meta-bit data are stored in memory 150 as shown in drawing 31 .
[0036] As shown in drawing 2 , it is foreknown that the IOT controller 160 is connected to a bus 114 through a bus interface 1610. A SunSPArcTM workstation is used in the gestalt of desirable operation of OOESS100. Thus, a bus 114 is SBus (S bus), and a bus interface 1610 may be designed so that it may work with SBus114. However, a bus interface 1610 may be designed so that it may work with the specific bus 114 which may be used also in any, such as the conventional personal computer, an en JIRIA ring workstation like SunSpArcTM, and a microcomputer.
[0037] As shown in drawing 2 , a bus interface 1610 offers 32-bit connection to a bus 114. Thus, a bus interface 1610 can input 4-byte WORD between each clock cycle. In the gestalt of desirable operation, a bus interface 1610 can be read by 64-byte burst by reading 16 4-byte

WORD continuously by the continuous clock cycle. Furthermore, the bus interface 1610 in the gestalt of desirable operation offers the direct memory address (DMA) to memory 150 through a bus 114.
[0038] If a bus interface 1610 receives the data of a 4-byte portion from memory 150, data will be distributed to one set of the five-set FIFO (FIFO) data register. These five FIFO contains the sample color channel FIFO 1620, the mask-data channel FIFO 1622, the fixed color channel FIFO 1624, the meta-bit channel FIFO 1626, and the command instruction channel FIFO 1628.
[0039] As shown in drawing 2 , the sample color channel FIFO 1620 consists of two banks of FIFO, and each bank of FIFO consists of two 512 address X 9-bit width of face FIFO. Thus, each FIFO of each bank of FIFO receives 1 byte of the 4-byte WORD received with the bus interface 1610. Furthermore, a bus interface 1610 generates the data of the bit of four additions, and one of the data of the bit of this addition is stored in each of four FIFO as 9th bit. Each of the bit of these excesses is used in order to carry out the flag of whether the data of the cutting tool relevant to it is an effective data, or it is an invalid data (a flag is set). Since the ejection of DMA data is made only in the word boundary, it sometimes happens that the actual data to the sampled image starts within WORD. In this case, a flag is set as an invalid, the cutting tool in the WORD preceded with initiation of actual data is not printed, and these cutting tools may be canceled.
[0040] The mask-data channel FIFO 1622 consists of a single 256 address X 16-bit width-of-face FIFO data register. The fixed color channel FIFO 1624 consists of one bank of two FIFO, and each FIFO of a bank consists of one 256 address X 9-bit width-of-face FIFO data register. Since each of a mask FIFO 1622 and a color FIFO 1624 can store 2 bytes per write cycle, data can be continuously offered to both a mask FIFO 1622 and the fixed color FIFO 1624 using single [4 bytes of] transfer of the data inputted into a bus interface 1610. Since the only 16-bit width-of-face internal bus connects a bus interface 1610 to a mask FIFO 1622, the fixed color FIFO 1624, and a command FIFO 1628 and the only 8-bit bus connects a bus interface 1610 to the meta-bit FIFO 1626, it is required to be used in order that two clock cycles may write one 4-byte imprint in a mask FIFO 1662, a color FIFO 1624, and a command FIFO 1628, and for four clock cycles to write a 4-byte imprint in the meta-bit FIFO 1626.
[0041] The meta-bit FIFO 1626 consists of a single 512 address X 8-bit width-of-face FIFO data register. The command instruction FIFO 1628 consists of a single bank of the 512 address X 8-bit width-of-face FIFO data register of a pair.
[0042] Each output of FIFO1620 thru/or 1628 is connected to multi-channel KOMBANA (coupler) 1630. The output from a mask FIFO 1622 is made serial. Thus, the only single bit connection is offered from a mask FIFO 1622 to the multi-channel combiner 1630. Similarly, the sample color FIFO 1620 and the fixed color FIFO 1624 are multiplexed by (2-to-1) from 2 to 1. Thus, the only 18-line internal connection is offered to the sample color FIFO 1620 to a multi-channel combiner. Each bank of a pair of 9-bit width of face FIFO outputs the data by turns. Similarly, only 9-bit width-of-face connection is offered between the fixed color FIFO 1624 and the multi-channel combiner 1630. The 9-bit each width of face FIFO of the fixed color FIFO 1624 outputs the data to a multi-channel combiner by turns.
[0043] As the meta-bit FIFO 1626 and the command instruction FIFO 1628 are indicated by contrast to be FIFO1620 thru/or 1624 to drawing 2 , the full width-of-face connection with the multi-combiner 1630 is offered, respectively.
[0044] The multi-channel combiner 1630 fully explained by the following in relation to drawing 3 combines the data from the sample color channel 1620 and the fixed color channel 1624 to the output of the single stream to data FIFO 1642 based on the bit map data received from the instruction received from the command instruction FIFO 1628, and the mask FIFO 1622.
[0045] Data FIFO 1642 consists of a 4K address X 9-bit width-of-face FIFO data register. Since a multi-channel combiner outputs data to data FIFO 1642 from a 8-bit width-of-face cutting tool, the 9th bit of data FIFO 1642 is used for storing the 1st bit (beginning) from the meta-bit FIFO 1626. Moreover, the multi-channel combiner 1630 is connected to the output meta-bit FIFO 1640. The output meta-bit FIFO 1640 consists of a 4K address X triplet width-of-face FIFO data register. The multi-channel combiner 1630 divides a 8-bit each meta-bit cutting tool

to at least two 4-bit width-of-face meta-bit nibbles. If the 1st bit of a 4-bit each nibble is outputted to data FIFO 1642, the 2nd to the 4th bit of a 4-bit each nibble will be outputted to the output meta-bit FIFO 1640. The number of the meta-bits outputted from a multi-channel combiner may actually be 1, 2, 4, or 8 depending on the number of the types of the object which should be distinguished, and the number of the level of each type of an object which should be performed.

[0046] Each command instruction defines the adjustable amount of the scan line only in relation to a single scan line so that it may be stated below. Therefore, since the output of a multi-channel combiner cannot be appropriately synchronized to the page and line of IOT, concurrency input port 1650 receives a page synchronous (sync) signal, a line synchronizing signal, and an IOT clock signal, and inputs into the multi-channel combiner 1638.

[0047] An output 1640 and FIFO 1642 is connected to the concurrency output port drivers 1652 and 1654, respectively. The output from these concurrency ports 1652 and 1654 is connected to the input port of IOT170.

[0048] Being received by IOT170 as data with which data FIFO 1642 should be processed for a print should be understood. However, this invention offers the print data of each cutting tool from data FIFO 1642 synchronizing by 1, 2, 4, or 8 meta-bit from the output meta-bit FIFO 1640, and being accompanied by it further. These meta-bits specify the processing which should be made to each data byte, before being printed. As explained, this optimal processing changes with each data byte depending on the object from which each data byte was extracted. As shown in drawing 34 , in the gestalt of suitable operation of the IOT controller 160, IOT170 has the image processing system 1710 containing the magnitude of the subsystem which is selectively controlled by the meta-bit data transmitted to IOT170 and by which object optimization was carried out. Thus, a meta-bit channel is made to process so that close may differ to each cutting tool's print data depending on the coming object.

[0049] For example, a meta-bit can be chosen among many halftone generators 1712. One halftone generator may be used with the tagged data byte, when it comes from an object like the text of the color which should use the frequency halftone generator between altitude. Other objects may be low spatial-frequency high color resolving halftone generators used with the data byte obtained from a photograph, pictures, etc. A meta-bit select is [to halftone generating like control of the halftone screen angle for every object rather than] possible for detailed control. In addition to selection of a halftone, deal with the problem resulting from having a dialog mutually with those edges of various objects. The object optimization color conversion subsystem 1714, the object optimization tone playback curvilinear modification subsystem 1716, the object optimization spatial filter subsystem 1718, and the object optimization trapping subsystem 1720, And the object optimization engine noise control subsystem which is going to control the problem of the various print systems which appear so that it may differ in a different object, In order to generate the adjustable result for which it depends on a list at an object like other object optimization image-processing subsystems, in an image processing system 1710, the image-processing subsystem of many additions may be controlled by the meta-bit.

[0050] IOT170 contains the laser power selected subsystem 1724 in an image processing system 1710. A meta-bit controls the laser selected subsystem 1724 of an image processing system 1710, and chooses it between different set points depending on the meta-bit relevant to each specific cutting tool of image data. Especially the object optimization laser power selected subsystem 1724 changes the set point of the laser power used in order to carry out the rendering of the cutting tool's image data based on whether the cutting tool of image data is in a halftone object, a text, or a line art object. In order to carry out the rendering of the object, the object optimization laser power selected subsystem 1724 enables selection between two or more set points so that the most suitable laser power may be offered.

[0051] For example, when directing that the cutting tool's image data is the portion of a text or a line art object, in order that a meta-bit value may control the object optimization laser power selected subsystem 1724 and may form the cutting tool's image data, a high laser power set point is used for a meta-bit value. Or when directing that the cutting tool's image data is in a halftone object, in order that a meta-bit value may control the object optimization laser power

selected subsystem 1724 and may form the cutting tool's image data, a low power set point is used for a meta-bit value.
[0052] It should be understood that the laser power relevant to each of these set points may be set up in the relation of other set points. for example, a low set point -- all -- others -- it is independently set up from a set point. Next, a high set point is set up more highly 10% than a low set point. Or a set point may be independently set up from each other.
[0053] Moreover, many printers are "write-in black" types, and he should understand that they expose the field of the existing media which may be developed marks like a text / line art. If light becomes higher, a mark will become thicker by exposure. Other printers are "white write-in" types which expose the field surrounding a mark, these exposure fields are not developed but an unexposed field is developed. Low exposure is required for creating a thicker mark in a white write-in system by the white field surrounding a text / line art.
[0054] Furthermore, when a print system has other synchronous data sources like an input scanner, since close mixes these data streams with the print data to which it comes from the IOT controller 160 through a data multiplexer 1730 in advance of a print, a meta-bit channel may be used. It may be used in order that the matching equipment may perform mixing in the IOT controller 160.
[0055] In the gestalt of implementation of instantiation of the equipment by above-mentioned this invention, a meta-bit chooses the usable line of a subset which controlled said two or more image-processing subsystems 1712-1738 by the base the whole object, and was specified in each of an object optimization image-processing module and the print multiplexer 1712-1730 by choosing between the one-set meta-bit mapping registers 1740 with which the output is loaded beforehand. For example, when 4 meta-bit is performed, they may be used, choosing them from 16 one-set registers. It may be the thing of all required sizes to be fully able to control these registers respectively and to be chosen from the subset of an image-processing subsystem and the print multiplexer 1712-1730 on the other hand. Thus, the semantics of each metabit value is completely programmable, and may be simply changed by changing the content of the register which it chooses. On the other hand, the register may be chosen from the perfect set of an image-processing subsystem and the print multiplexer 1712-1730 in order to perform a rendering with the highest possibility to the specific type of the object tagged by the meta-bit.
[0056] It should be understood that various image-processing subsystems and print multiplexers 1712-1730 may cover the whole object optimization print system by turns, and may appear on many of other points. For example, after an object type is determined, an object optimization image-processing subsystem and the print multiplexer 1712-1730 can be arranged in IOT170 as some IOT controllers 160 at any time to object optimization ESS 100 like [within the PDL decomposition means 130 and the IOT command instruction generating means 140], so that it may be illustrated. Moreover, these subsystems can be distributed to the whole system.
[0057] Furthermore, these object optimization image-processing subsystems and the print multiplexer 1712-1730 may be performed in hardware, software, or these two combination. In all cases, the subsystem has chosen the data based on the type of an object which a different procedure and different they are processing. Therefore, a different processing result to a different object type is generated.
[0058] Moreover, the meta-bit information connected with a measurement calibration print is used for the color measurement device 190, with the object optimization base, the image-processing subsystem 1712-1722 can be adjusted automatically, and it can stabilize it. For example, as shown in drawing 35 , the output color measurement device 190 which measures a calibration print outputs the signal which directs the actual condition of a calibration print. Based on the output signal from meta-bit information and an output color measurement device, the object optimization output correction subsystem 1732 adjusts one tone playback curvilinear (TRC) look-up table from the set of the whole in the object optimization tone playback curvilinear subsystem 1716. Since the object optimization output correction subsystem 1732 is controlled by the tag encoded by the meta-bit channel, the corrections performed to an image processing system function (function) differ to a different object.
[0059] Drawing 3 shows the gestalt of implementation of one instantiation of multi-channel 1630.

As shown in drawing 3 , it connects with the meta-bit unpacker 1631, and the meta-bit FIFO 1626 changes a 8-bit width-of-face meta-bit into at least two 4-bit width-of-face meta-bit nibbles. The output from the meta-bit unpacker 1631 is connected to each of four a color / meta-bit registers 1634-1637. The output from the meta-bit unpacker 1631 depends on the load enabling (LE) signal outputted with FIFO control, and the command / mask channel processor 1680, and is loaded to a color / meta-bit register 1634 thru/or one of the 1637. Similarly, the data input from the fixed color channel FIFO 1624 is offered four a color / meta-bit registers 1634 thru/or 1637, and is loaded to one of a color / the meta-bit registers based on a load enable signal.

[0060] The 18-bit width-of-face input data from the sample channel FIFO 1620 is inputted into sample channel unpacker 1632A, 2-byte width-of-face sample color data and 2-bit "validity-cutting tool" data are changed by the single cutting tool output of two pieces, and by outputting them to internal sample FIFO1632B continuously, those corresponding "valid bytes" are canceled, when directing that this cutting tool is invalid.

[0061] The direct input of the 2-byte lateral-spreading force from the command instruction FIFO 1628 and the 1-bit lateral-spreading force from a mask FIFO 1622 is carried out to a command / mask channel processor 1680. Moreover, the page synchronization, line synchronization, and IOT clock which are received from IOT through concurrency input port 1650 are inputted into a command / mask channel processor 1680. A color / meta-bit multiplexer 1633, the output multiplexer 1639, and four read-out signals to FIFO 1620-1628 are outputted for a command / mask channel processor 1680 control signal. Four read-out signals consist of a color / meta-bit read-out (RD) signal, sample read-out signals, mask read-out signals, and command instruction read-out signals. Shortly after corresponding FIFO 1620-1628 receives one of the signals of these from a command / mask channel processor 1680, corresponding FIFO (an unit or plurality) will read the next data on a corresponding channel. FIFO (an unit or plurality)1620 thru/or 1628 fill them up with the data by which reading appearance was carried out from the channel data structure (see drawing 35) which prevented dry cleaning (dry) running by the bus interface 1610, and was stored in memory 150 according to the calculated priority.

[0062] Based on the command instruction received from the command channel FIFO 1628, a command / mask channel processor 1680 generates load enable signal LE to one to which a color / meta-bit register 1634 thru/or 1637 correspond. If it enables one of a command / the mask channel processors 1680, it loads the 9-bit width-of-face fixed color data from the fixed color FIFO 1624, and the meta-bit data to 4 bits from the meta-bit unpacker 1631 simultaneously. Or color data may be independently loaded from a meta-bit by the separate command. However, since they are loaded simultaneously continuously, as for a color / meta-bit register 1634 thru/or 1637, it is desirable that it is a register according to individual always [both] referred to.

[0063] Moreover, a command / mask channel processor 1680 generates a control signal to a color / meta-bit multiplexer 1633, in order to choose the color / meta-bit register 1634 which should be inputted into the output multiplexer 1639 and the meta-bit register 1638 thru/or one of the 1637. Furthermore, a command / mask channel processor 1680 generates a control signal to the output multiplexer 1639, in order to choose between the output from a color / meta-bit multiplexer 1633, and the output from internal sample color FIFO1632B. The output multiplexer 1639 outputs the 8-bit width-of-face color data outputted by a color / meta-bit multiplexer 1633 and the metadata of the 1st bit. This is inputted into data FIFO 1642. Simultaneously, the meta-bit register 1638 stores the 2nd - the 4th meta-bit, and outputs them to the output meta-bit FIFO 1640. It should be understood that it is unnecessary in the thing [as / in the gestalt of desirable operation] containing all the 4-bit width-of-face meta-bit nibbles. Or the number of the meta-bits offered to IOT may be 1, 2, 4, or 8. Thus, the gestalt of implementation of instantiation shown in drawing 2 and 3 should not be interpreted as what restricts the number of the meta-bits offered to IOT to 4.

[0064] Drawing 4 shows internal functional block of a command / mask channel processor 1680. In drawing 4 , only the control line is shown except for the command channel line and the mask data channel line. In the command / mask channel processor 1680, two configuration registers

1681 and 1684, a register CSR 0, and CSR1, are prepared, respectively. The 1st configuration register 1681 offers control of the feature (function) shown in a table 1.
[A table 1]

ビット	ハイ	ロー
8	将来の使用のためにリザーブ	将来の使用のためにリザーブ
7	白=FF	白=00
6	ノーマルモード	診断モード
5	レジスタ読出し	FIFOs読出し
4	FIFOモードに対してレジスタをオン	FIFOモードに対してレジスタをオフ
3	メタビットFIFOを使用	カラーFIFOの9番目のビットを使用
2	メタビットアンパック1をオン	メタビットアンパック1をオフ
1	メタビットアンパック0をオン	メタビットアンパック1をオフ

構成／ステータスレジスタ0(CSR0)ビット割り当て

[0065] The 1st bit 1 and bit 2 of the configuration register 1681 define 2 (need is accepted or it is 3 or 4) meta-BITTOAN packing method used by the meta-bit unpacker, in order to unpack a meta-bit. That is, when the method of the addition for determining how the cutting tool of meta-bit data is decomposed to meta-bit PAKKETTO is required, it may be used in order that the 1st and 2nd bits may offer [both] the type to four of meta-BITTOAN packings.
[0066] When the 1-bit width-of-face meta-bit channel of one ** is offered to IOT, the bit 3 of the 1st configuration register 1681 is used in order to make the meta-bit FIFO disenabled (an activity is impossible). The 1st bit 4 and bit 5 of the configuration register 1681 are used by the diagnostic mode, and read data from FIFO and an internal register. The bit 6 of the 1st configuration register 1681 is used in order that a multi-channel combiner may direct normal mode or the diagnostic mode. The bit 7 of the 1st configuration register 1681 is used in order that that all white (for example, natural background color of paper) is offered may direct the data byte which consists of 0, or the data byte which consists of 1 altogether. The 8th bit of the 1st configuration register 1681 is not performed.
[0067] When the bit 6 of the 1st configuration register 1681 is a low, the multi-channel combiner 1630 is the diagnostic mode. In the diagnostic mode, all of the internal register of the multi-channel combiner 1630 may be examined. For example, the content of color / meta-bit register 1634 thru/or 1637, and sample FIFO1632B is examinable on real time. By setting the diagnostic SEL line shown in a table 2 to the value of 0-4, a SEL line can send the value of the selected register to data FIFO 1642 compulsorily. As shown in a table 2, the value 5-14 of a SEL line is used for read-out of other registers of the multi-channel combiner 1630, and writing for the further diagnostic information.
[A table 2]

SEL	アクセスされるレジスタ
0	ビデオFIFOへのカラー/メタビットレジスタ0
1	ビデオFIFOへのカラー/メタビットレジスタ1
2	ビデオFIFOへのカラー/メタビットレジスタ2
3	ビデオFIFOへのカラー/メタビットレジスタ3
4	ビデオFIFOへのダンプ#Bレジスタ
5	リードバックバスに配されるSR0
6	リードバックバスに配されるSR1
7	リードバックバスに配される又は書込まれるCSR0
8	リードバックバスに配される又は書込まれるCSR1
9	リードバックバスに配されるメタビット出力レジスタ
10	リードバックバスに配される又は書込まれるビデオデータカウンタロー (0:7)
11	リードバックバスに配される又は書込まれるビデオデータカウンタハイ (8:12)
12	リードバックバスに配される又は書込まれる先端白カウンタ (0:7)
13	リードバックバスに配される又は書込まれる後端白カウンタ (0:7)
14	リードバックバスに配されるフルフラグレジスタ
15	将来使用のためにリザーブ

診断SEL値に対するレジスタアクセスデコード表

[0068] When the bit 6 of the configuration register 1681 is yes (namely, normal mode), the multi-channel combiner 1630 generates the Normal print data, and it is transmitted to it to data FIFO 1642.
[0069] A table 3 shows bit assignment of the 2nd configuration register 1684. A bit 7 and a bit 8

provide an IOT interface with flexibility by determining the polarity of an IOT interface signal. A bit 5 and a bit 6 offer diagnostic exchange. Bits 1, 2, 3, and 4 are not performed with the gestalt of implementation of this instantiation.

[A table 3]

ビット	ハイ	ロー
8	Ｐ同期立下り	Ｐ同期立ち上がり
7	Ｌ同期立下り	Ｌ同期立ち上がり
6	偽Ｐ同期オン	偽Ｐ同期オフ
5	偽Ｌ同期オン	偽Ｌ同期オフ
4	将来使用のためリザーブ	将来使用のためリザーブ
3	将来使用のためリザーブ	将来使用のためリザーブ
2	将来使用のためリザーブ	将来使用のためリザーブ
1	将来使用のためリザーブ	将来使用のためリザーブ

構成／ステータスレジスタ１(CSR1)ビット割り当て

[0070] If drawing 4 is referred to again, a command / mask controller 1686 will receive a 16-bit command instruction and a 1-bit mask channel input. Normal / repeat mode bit sets tables 4 and 5 to Normal -- having (that is, a bit 15 being set to 0) -- the bit assignment to a 16-bit command instruction is shown.

[A table 4]

コマンドビット	フィールド記述子
0	RepCnt0
1	RepCnt1
2	RepCnt2
3	RepCnt3
4	RepCnt4
5	RepCnt5
6	SCRes0
7	SCRes1
8	SrcB0
9	SrcB1
10	SrcA0
11	SrcA1
12	Cntr10
13	Cntrl1
14	LdColor
15	Repeat/Normal

ノーマルモード（ビット15=0）

[A table 5]

	LdColor	コマンド
	0	カラーをロードしない
	1	SrcB CC/メタビットレジスタをFIFOsからロード 注:1）"メタビットモード"を使用の場合、 メタビットがアンパッカーからロードされる。 注:2）カラー/メタビットレジスタは、これがカラー/メタビットレジスタ0乃至2に対するこのコマンドの第1のクロックサイクル又はカラー/メタビットレジスタ3のこのコマンドのあらゆるクロックサイクルである時、ロードされる。
Cntrl1	Cntrl0	コマンド
0	0	常にSrcB
0	1	常にSrcA
1	0	マスクがSrcAとSrcBとの間で選択
1	1	マスクがSrcBとSrcAとの間で選択
SrcA1	SrcA0	コマンド
0	0	SrcAがカラー/メタビットレジスタ0を選択
0	1	SrcAがカラー/メタビットレジスタ1を選択
1	0	SrcAがカラー/メタビットレジスタ2を選択
1	1	SrcAがサンプルを選択
SrcB1	SrcB0	コマンド
0	0	SrcBがカラー/メタビットレジスタ0を選択
0	1	SrcBがカラー/メタビットレジスタ1を選択
1	0	SrcBがカラー/メタビットレジスタ2を選択
1	1	SrcBがカラー/メタビットレジスタ3を選択
SCRes1	SCRes0	コマンド
0	0	サンプルクロックを4で割る
0	1	サンプルクロックを3で割る
1	0	サンプルクロックを2で割る
1	1	サンプルクロックを1で割る
	RepCnt0-5	コマンド
		ノーマルコマンドリピートカウント

ノーマルコマンドデコード表

[0071] In normal mode, as shown in a table 4, a bit 0-5 expresses 6 bit-count value. As for counted value, that it is the pixel which this command follows directs how many it is effective. Depending on a value [as opposed to bits 6 and 7 in a table 5], a sample clock is divided by 1, 2, 3, or 4. By breaking a sample clock, when the resolution of an output image carries out the repeat output of the current color of internal sample FIFO1632B, it may change. Namely, when both the values of the bits 6 and 7 as which a crack" command is chosen by 4 in "sample clock are 0, the 4 times repeat output of the internal sample FIFO1632B is carried out for every clock cycle. It follows, for example, the printer of 400 spots / inch (spi) resolution does not have the need, and processing of in any way an addition can print a sample map by 400spi(s), 200spi, 133spi, and 100spi(s).

[0072] A color register-select B value is expressed, and, as for bits 8 and 9, bits 10 and 11 express a color register-select A value. As shown in a table 5, as for a color register-select A value and a color register-select B value, any of a color / meta-bit register 1634-1637 (or internal sample FIFO1632B) direct whether it is passed by the combiner to data FIFO 1642. . which a color register-select A value chooses between a color / megabit registers 1634, 1635, and 1636, or the internal sample FIFO, and a color register-select B value chooses between a color / meta-bit register 1634-1637 -- there are two selection values A and B, and, thereby, a 1-bit channel is used for the change between them. This mode is specified by bits 12 and 13, and it is used in order to switch promptly between two registers specified without the instruction overhead of any addition by A and B.

[0073] Bits 12 and 13 express a 2-bit combiner control value. A combiner control value directs

the color data which should be outputted to data FIFO 1642 by the multi-channel combiner 1630 to the following n pixels directed with 6 bit-count value. As shown in a table 5, it is directed whether a combiner control value comes from the combination of the register with which the color which should be outputted is chosen by a color register-select A value or the color register-select B value, or its two registers which are further controlled by the mask channel.
[0074] Finally, as shown in a table 5, a bit 14 directs whether the following color data outputted from the color channel FIFO 1624 should be loaded to one of a color / the meta-bit registers 1634-1637 by the color register-select B value.
[0075] In actuation, if a new normal mode command is inputted from a command channel FIFO 1628, the load color bits 14 will be decoded, in order that the following 8-bit color data byte stored in the color channel FIFO 1624 may determine whether it should be loaded to the color / meta-bit register 1634-1637 directed by color register-select B bit 8 and 9. When color register-select B bit 8 and 9 directs a color / meta-bit register 1634-1636 (a color / meta-bit register 0-2), a color is loaded from the color channel FIFO 1624 only by the 1st clock cycle of a command.
[0076] However, color register-select B bit's 8 and 9 directions of a color / meta-bit register 1637 (a color / meta-bit register 3) load new fixed color data to a color / meta-bit register 1637 from the color channel FIFO 1624 for every clock cycle of this command. Thus, a fixed color channel may be actually used as a low-speed sample channel. In the high image of some complexity, since the fixed color data in a scan line change promptly (that is, 1 - 8-pixel all are changed), they are that, and they generate a new command word for every new color, and it stores, and transmits, and processing required in order to decode becomes superfluous, consequently a cutting tool map expression is faced and it becomes negative compression.
[0077] This effective negative compression may be avoided, when the string of a fixed color is offered to the fixed color channel FIFO 1624, one fixed color cutting tool of the string of a color FIFO 1624 to a fixed color is loaded to read-out and it loads it to a color / meta-bit register 1637 for every clock cycle. Since the fixed color channel FIFO 1624 can load 2 bytes at once as in the sampled data channel FIFO 1620 not at 4 bytes but at 1 time, as for using the fixed color channel FIFO 1624 in this way, the throughput of a fixed color channel becomes lower than the throughput of a sample data channel. However, the this "slow sample" method avoids a required overhead, when generating the DMA pointer used for a regular sample data channel (explained in full detail below). Thus, to the short stretch of the fixed color which changes promptly, a this "slow sample" channel is not used to a big image, although it is the most useful.
[0078] Next, the combiner control bits 12 and 13 are decoded in order to determine whether it is used in order that the mask data from a mask channel may control the data flow to data FIFO 1642 further from the multi-channel combiner 1630, while determining any of the color selection register banks A and B are used, in order to determine any shall be used as a color data source between a color / meta-bit register 1634-1637 (or internal sample FIFO1632B). Of course, if it directs that the load color bits 14 and color selection register B bit 8 and 9 use a color / meta-bit register 1637 as a "slow sample" color channel, control bits 12 and 13 must direct to use a color register-select B bank. On the other hand, in order that the command bits 12 and 13 may direct a color data source, when it directs to use the color register-select bank A, the load color bits 14 and color register-select bank B bit 8 and 9 may be used in order PURIRODO [the following color data word which is not directed by 10 and 11 A bits of color register selects, and is outputted to one of a color / the meta-bit registers 1634-1637 from the fixed color channel FIFO 1624]. Thus, PURIRODO [the following fixed color data word / one of a color / the meta-bit registers].
[0079] Next, it is decoded in order to determine any shall be chosen depending on any of the register-select banks A and B are directed by the combiner control bits 13 and 14 between a color / meta-bit register 1634-1637 (or internal sample FIFO1632B) as a color data source with which 10, 11, or color register-select B bit 8 and 8 is outputted A bits of color register selects. Next, if the combiner control bits 12 and 13 direct the color register-select bank A and 10 and 11 direct to use the data stored in internal sample FIFO1632B A bits of color register selects, the sampled color resolution bits 6 and 7 will be decoded in order to determine the factor which

divides a sample clock. Next, directions of that the combiner control bits 12 and 13 control an output color further using a 1-bit mask channel switch the output to data FIFO 1642 between the color register (or the sample FIFO) chosen by 10 and 11 A bits of selections, and the color register chosen by selection B bit 8 and 9. Finally, a command instruction current in the repeat count bit 0-5 is decoded in order to determine how many a pixel is effective.

[0080] Therefore, a color / meta-bit register load selection 1685 is controlled by the value of the load color bits 14 and color register-select B bit 8 and 9 to be shown in drawing 4 . Similarly, a color / meta-bit multiplexer selection controller 1689, and the output multiplexer selection controller 1692 are controlled by combination of 10, 11, and a mask channel as mentioned above the command control bits 12 and 13, color register-select B bit 8 and 9, and A bits of color register selects.

[0081] However, if the repeat mode bit 15 is set to a high, the gestalt the command bit of a command instruction is instructed to be to tables 6 and 7 will be taken.

[A table 6]

コマンドビット	フィールド記述子
0	RepCnt0
1	RepCnt1
2	RepCnt2
3	RepCnt3
4	RepCnt4
5	RepCnt5
6	RepCnt6
7	RepCnt7
8	RepCnt8
9	RepCnt9
10	RepCnt10
11	RepCnt11
12	RepCnt12/Mask Scanline Disable
13	RepType0
14	RepType1
15	Repeat/Normal

繰り返しモード（ビット15=1）

[A table 7]

RepType1	RepType0	コマンド
0	0	最後のﾉｰﾏﾙｺﾏﾝﾄﾞを繰り返せ
0	1	白出力：先端白ﾃﾞｰﾀのために使用。繰り返しｶｳﾝﾄが0に等しくあるべきという点に注意。
1	0	ﾗｲﾝ端(End of Line)：後端白ﾃﾞｰﾀのために使用。繰り返しｶｳﾝﾄが0に等しくあるべきという点に注意。
1	1	ﾍﾟｰｼﾞ端(End of Page)
	Rep12	コマンド
	0	注：ﾗｲﾝ端がこのｺﾏﾝﾄﾞからﾃﾞｺｰﾄﾞされると、Rep12は、Mask Channel Scanline Disable; もしこのﾋﾞｯﾄがﾛｰの時、Mask Channelは次のｽｷｬﾝﾗｲﾝに対して使用不能である。
	1	ﾗｲﾝ端の時、Mask Channelは、次のｽｷｬﾝﾗｲﾝに対して使用不能である。
	RepCnt0-12	コマンド
		ｺﾏﾝﾄﾞ繰り返しｶｳﾝﾄ(Command Repeat Count)を繰り返せ。

繰り返しコマンドデコード表

[0082] As shown in tables 6 and 7, the command bit 0-12 expresses a repeat count. Since the repeat count in repeat mode offers the range of a repeat count 128 times in normal mode,

repeat mode is used when it has the same color data or the same object like the image with which the very broad structure of a scan line was scanned, or a big fixed color field.
[0083] Finally, bits 13 and 14 express the repeat type of a repeat mode command instruction. When a repeat type is normal, a front command is repeated to the number of pixels of the addition directed by the repeat count bit 0-11. In this case, the command register 1682 before being shown in drawing 4 is used. A repeat command type's directions of the edge of head white data or after [a line] white data load one single white cutting tool to data FIFO 1642 by the white (transparence) generator 1690. This white (transparence) generator 1690 is set to white correction value with the configuration register 1681. If a repeat command type directs the edge of the page, this command will suspend activation of the further command until a circuit is reset.

[0084] Only when the repeat type bits 13 and 14 direct a line type edge, a bit 12 should be redefined so that it may be mask scan line enabling / disenabled bit, and being used since the bit turns on or turns off a mask channel to the following scan line should be understood. Thereby, only the scan line which has these related mask data is allotted to a mask channel, and compresses the data which needs it for a mask channel. This is shown in a table 7.
[0085] The repeat type bits' 13 and 14 directions of an output white type or line type edge should also understand that a repeat count is not used. Instead, the white space of the start of a line and an end is the special case of video-data generating with the advantage which increases the amount of time amount which can be used since actual page data is generated. PURIRODO [three counters are attached in the IOT interface controller 1691, and / data with these exact at the beginning of each color separation] since a head and back end white (transparence) data are generated. The head white counter 1694 is used in order to count the initiation margin of each scan line. A count will be started if a line synchronizing signal with new line and page synchronous (sync) handler 1688 is received from IOT. While the head white counter 1694 has counted, the IOT interface controller 1691 carries out the repeat output of the single white cutting tool who made it activity impossible to read-out enable [of data FIFO 1642], and was allotted to data FIFO 1642 by the head white repeat command. Repeat reading appearance of this white cutting tool is carried out by IOT170. If the head white counter 1694 amounts to 0, the video-data counter 1695 starts a count, and the IOT interface controller 1691 will make data FIFO 1642 and the output meta-bit FIFO 1640 enable (usable), and will pour those data to IOT170 under the synchronization from the IOT controller 1691. If the video-data counter 1695 amounts to 0, the back end white counter 1696 will start a count, and the IOT interface controller 1691 will carry out repeat reading appearance of the single white cutting tool allotted to data FIFO 1642 by IOT170.
[0086] The IOT interface controller 1691 operates independently from the portion made to fill up with the data of FIFO control, and the command / mask channel processor 1680, and the output meta-bit 1642 and FIFO 1640. The IOT interface controller 1691 receives a clock signal, a page synchronizing signal, and a line synchronizing signal from IOT170 through a line and the page synchronous handler 1688, and returns a return clock signal and a read-out data signal to IOT170. As long as it has the effective data with which the multi-channel combiner 1630 was filled up with data FIFO 1642 and the output meta-bit FIFO 1640, the IOT interface controller 1691 is used in order to synchronize read-out of the data by exact time amount by IOT.
[0087] As shown in drawing 5 , a bus interface 1610 has burst FIFO 1611, the sample channel control 1612, the slow channel control 1613, a register 1614, a decoder 1615, buffer memory 1616, the channel mediation logic 1617, and the card controller 1618.
[0088] Basic actuation of a bus interface 1610 is searching PAKKETTO on the bus which uses a direct memory access controller (DMA), and writing them in one of the five FIFO channels 1620-1628. A bus interface 1610 is once programmed in front of each page, and since it is filled up with the FIFO channel 1620-1628 next, it operates independently. The channel mediation logic 1617 is used in order to determine the priority with which each of the FIFOF channel 1620-1628 is filled up. This is based on the signal from each of FIFO 1620-1628 generated when it is almost empty, programmable" weight" to each channel, and the interleave option to three of these channels. Weight directs the count of the line which remains as priority with the present highest

channel. The mediation method of the gestalt of operation of the 1st of a bus interface 1610 is based on three subsets of priority. The priority subset 1 is the always highest priority, and it includes drawing of a new sample and a mask channel pointer so that it may be explained in full detail later. Priority carries out alternation of the priority subsets 2 and 3. The priority subset 3 of the priority subset 2 is a round robin drawing command including sample channel data, and it takes out a mask, a color, a meta-bit, and command channel data.
[0089] The reason for a background of this mediation method is as follows. Although a pointer (subset 1) is small and there are not, as stated below, it is important drawing. [many] Sample channel data (subset 2) is the raw cutting tool who may demand the biggest bandwidth through a system. Other channels are called a "slow channel." The reason is that it has width of face with few those channels FIFO 1622-1628 than 32-bit width of face. If they are taken out, it is required to write the taken-out WORD in burst FIFO 1611, and, thereby, the slow channel control 1613 can unpack them on the FIFO channel 1622-1628 according to those individuals in parallel. Between this actuation, other burst ejection may occur in parallel to the sample channel FIFO 1620.
[0090] The burst transmission size of drawing is controllable to each channel. One of the registers 1614 stores the burst size to each channel. Moreover, there is an one-set register of the register 1614 which stores the address in the memory 150 which the data to each channel starts. Since memory size is saved, the offline processing (indirection) of 1 level is used for a mask channel and a sample channel in data ejection. Instead of using a single address-pointer register, each of a mask and a sample color channel has three registers. The 1st thing points at the address in the memory 150 in which the table (table) is stored. These each of a pair of specifies the portion of the memory 150 by which a next sample map or a next bit map section is found including the list of the address / size pairs. [table / this] The sample channel control 1612 takes out the next address to the sample address register of a register 1614, and takes out the size of the block of drawing and sample data to the sample size register of a register 1614. Next, it can be taken out from the portion of memory 150, and it takes out the following address / size pair in a sample data table after that until the data of an amount with an exact bus interface 1610 is taken out. A mask channel is processed similarly.
[0091] Since it generates continuously in the word boundary, in the case of the sample data which the data of the 1st actual valid byte generates in WORD, drawing is possible. Thus, the sample channel control 1612 compares the word address in the memory 150 actually taken out in the byte address in the memory 150 which was going to be taken out, and must be able to tag it to 3 bytes of the 1st WORD taken out with the "invalid" tag. Sample channel unpacker 1632A does not load these invalid cutting tools to internal sample FIFO1632B, and can cancel them later.
[0092] Other pairs of a bus interface 1610 contain the decoder 1615 which processes the buffer memory 1616 for making a data rate equal, the card controller 1618 which processes a low-level S bus (Sbus) card function, S BASURI QUEST address mapping, and decoding. One important function of a decoder 1615 is offering the interface to the serial port of IOT170 which transmits status information and can receive.
[0093] Drawing 6 shows the gestalt of implementation of instantiation of the 2nd of the IOT controller 260. As shown in drawing 6 , the bus interface 2610 with same bus interface 1610, essential actuation, and configuration receives a 4-byte burst from a bus 114, and outputs a 32-bit data stream to a compressor (compressor) / decompressure (compression discharge machine) controller 2650, and the integrated combiner / channel FIFO 2630. An integrated combiner / channel FIFO 2630 unifies the multi-channel combiner 1630 with a data channel FIFO 1620-1628. A compressor / decompressure controller 2650 is controlled by meta-bit information transmitted from memory 150 while it enables the activity of the data compressed when transmitting data to the IOT controller 260 from memory 150. A compressor / decompressure controller 2650 enables meta-bit information to direct which type of compression was applied to the block with which data was received in this way.
[0094] Therefore, a data compression can be optimized based on an object type (that is, is data other data types which may be compressed the optimal using the color picture data compressed

the optimal using the "JPEG" (joint photographic expert group) technique, white / black bit map data and run length encoding, or a binary compression technique like CCITT?). Therefore, based on the object type of the data inputted into a compressor / decompressure controller 2650, data is outputted to a sample data compressor / decompressure 2620, or the binary sample data compressor / decompressure 2640. Furthermore, the scanner interface 116 (you may be an interface to other devices like a digital camera) can be attached in a compressor / decompressure controller 2650, and it is real time, and thereby, the on-the-fly data from a scanner is obtained and compressed, it is transmitted to a compressor / decompressure controller 2650, compression discharge is carried out, and it may be inputted into the suitable channel of an integrated combiner / channel FIFO 2630. On-the-fly insertion of the portion of an image is enabled by this through the scanner or the object of other equivalence like the scanner of a copying machine, or an integrated copying machine / printer, and thereby, on-the-fly data does not need to change PDL description of a page image, and may be combined under control of a meta-bit channel to a page image. Furthermore, thereby, on-the-fly optimization of the portion of an image is enabled through a scanner or other equivalent devices. For example, if the rendering of the portion of an image should be carried out using the laser beam of higher level, an operator uses the increased exposure level set point, and can identify the rendering of which portion of an image should be carried out.

[0095] Once compression discharge of a sample or the binary data is carried out, it will be inputted into an integrated combiner / channel FIFO 2630, and it will operate, as substantially mentioned above about drawing 2 -4.

[0096] Furthermore, since the IOT controller 260 contains the color space converter (transformer) 2670, sample color data and fixed color data can be arranged on the optimal color space, before it is stored or transmitted. As shown in drawing 6 , an integrated combiner / channel FIFO 2630 outputs 24-bit WORD (per color three colors X 8 bits) to the color space converter 2670, and this converter changes 24-bit 3 color data into 4 bytes of color data, next that data is returned to an integrated combiner / channel FIFO 2630 as 4 bytes showing C, Y, M, and K color detached core. Moreover, color data may be directly inputted into the color space converter 2670 from a bus interface 2610. It may be used in order to choose the optimal color space conversion to the specific object type to which meta-bit data control the color space converter 2670, and current data relates in both cases. As a portion of object optimization color space conversion, an optimal tone playback curve (TRC) may be chosen using a meta-bit. Next, the integrated combiner / channel FIFO 2630 which operates as mentioned above in relation to drawing 24 output output color data and a meta-bit to IOT170. According to the capacity and speed of IOT170, the last data may be outputted in a 8-bit cycle plus 1per output-4 meta-bit, or a 32-bit cycle plus 1per output-4 meta-bit.

[0097] In the gestalt of implementation of instantiation of the 3rd of the IOT controller shown in drawing 7 , a compressor / decompressure 3650 is used only in order to control a sample data compressor / decompressure 3620. The binary sample data compressor / decompressure 2640 of drawing 6 were replaced with more advanced structure. In the gestalt of implementation of this instantiation, a binary compressor / decompressure 3640 is in any of an adaptation compressor / decompressure, and a high-speed CCITT compressor / decompressure. However, this IOT controller 360 operates, as generally mentioned above about the gestalt of the 1st of an IOT controller, and the 2nd suitable operation.

[0098] Finally, drawing 8 shows the gestalt of implementation of instantiation of the 4th of the IOT controller 460. In the gestalt of implementation of this 4th instantiation, an integrated combiner and a channel FIFO 4630 output three colors (24-bit WORD) or four colors (32-bit WORD) to the 1st color space converter 4670 and the 2nd color space converter 4675. Since two of 4 color detached cores outputted to IOT170 are generated, each of the 1st color space converter 4670 and the 2nd color space converter 4675 is used. Generally, each of a color space converter operates at speed twice the speed [the combiner integrated, a channel FIFO 4630, and] of packers 4672 and 4677.

[0099] For example, it is the 1st clock cycle, and the 1st color space converter 4670 outputs a 8-bit C color detached core data byte, it is the 2nd clock cycle and outputs a 8 bit M color

detached core cutting tool. Similarly, by the 1st clock cycle, the 2nd color space converter 4675 outputs a 8 bit Y color detached core cutting tool, is the 2nd clock cycle and outputs a 8 bit K color detached core cutting tool.
[0100] By the 1st clock cycle, the 1st color space converter 4670 outputs 8-bit C data to a packer 4672, and the 2nd color space converter 4675 outputs 8 bit Y data to a packer 4677. Similarly, by the 2nd clock cycle, the 1st color space converter 4670 outputs 8 bit M data to a packer 4672, and the 2nd color space converter 4675 outputs 8 bit K data to a packer 4677. Since packers 4672 and 4677 operate in the one half of the speed of the color space converter 4670 and 4675 ** next, they output 16 bit data unified with IOT170 from the 1st and 2nd color space converters 4670 and 4675 in parallel, and offer four color detached cores simultaneously as four 8-bit WORD altogether. Thus, the larger range of IOT170 can use it by this system, and this system contains the one pass, and 4 color copying machines / printer which uses four different print drums, in order to be the single pass of a copy sheet and to form all 4 color detached cores, and C, M, Y and K through IOT.
[0101] The following drawings show the data structure stored in the PDL decomposition means 130, IOT data, the command instruction generating means 140, and memory 150. Memory 150 contains the RAM portion 151 and the non-volatile portion 152. The non-volatile portion 152 may consist of either of the memory devices of other equivalence which offer a hard disk, a dismountable tape, a floppy disk, an optical disk, a flash memory, and the non-volatile data storage over a long period of time. As shown in drawing 9 , after starting at step S10, a document is created at step S20 by document creator like an operator with a graphical artist or the technology of other desktop publication programs.
[0102] Once a creator ends document creation at step S20, the print data for IOT will be prepared at step S30 using object assignment or object optimization compression, and a rendering technique. Preparations of the print data in step S30 are made by the PDL decomposition means 130, IOT data, and the command instruction generating means 140. Preparation of print data stores it in the RAM portion 151 of memory 150 using the data structure shown in drawing 28 -31.
[0103] Next, after print data is prepared at step S30, in step S40, it is determined whether the data structure of the RAM portion 151 of memory 150 should be stored in the non-volatile portion 152 of memory 150. **** with the storing step S50 a document creator may specify the storing step S50, and big [a print system] -- you may require that the resource of a complicated document should be saved. Decision of saying [that the data structure stored in the RAM portion 151 of memory 150 should be memorized] stores a nonvolatile memory portion in the compressed page at step S50. After determining that the compressed page to this document stored by S50 should be printed, this requires inevitably that the page compressed as in step S60 should be searched, when a creator or a print system copies them to the RAM portion 151 from the non-volatile portion 152 of memory 150. Thus, the print data prepared at step S30 is outputted to step S70 irrespective of whether it was stored at step S50 and was again called at step S60, or it was directly transmitted by step S40.
[0104] Moreover, IOT170 is not driving actively step S20-S50, step S110-130, and those substeps described later, therefore since real-time constraint does not restrict these steps, it should be understood that it is not necessary to perform on real time. Reversely, since IOT170 is driving step S70-S100 actively, it should be understood that it should perform on real time. Therefore, processing of the data in real time and the page to IOT170 on which failure of offer is printed by IOT are made into incorrectness. Depending on the type and capacity of the nonvolatile memory portion 152, step S60 does not have to be carried out, even if it performs on real time.
[0105] In step S70, the print data prepared at step S30 is combined and printed using object assignment or object optimization compression discharge, and a rendering technique. Simultaneously, at step S80, real-time data is captured and synchronizes with the print data prepared at step S30. The real-time data of step S80 is captured from other sources like other devices which generate a scanner, the scanner portion of a copying machine, a digital camera, a remote computer, or data, and can be transmitted to the IOT controller 160 on real time.

[0106] Next, at step S90, a creator or a print system determines whether this print actuation is a calibration print. A creator or automatic calibration processing of IOT170 is step S90, and if it determines that this print is a calibration print, a print page will be automatically measured on real time in step S100, in order to determine whether object optimization rendering accommodation needs to correct which rendering of the object of a document. For example, the IOT controller 160 outputs a predetermined test document to IOT170 during the calibration test of one specific type. This test document includes the test patch by which a rendering is carried out, in order to simulate an object of a specific type like a pictures target or a photograph-(sampled) object. the inside of IOT170 after IOT170 prints a test document -- or a mounting **** sensor measures the various colors of a test document to it, and provides it with the measurement data to the IOT controller 160. Being printed on a test document to the type of the actual color and object which were printed as directions by measurement data compares the color meant, and it adjusts the IOT controller 160. In this way, the tone playback curve drift of IOT170 produced by change in temperature, humidity, or other environmental factors may be corrected with the object optimization base by [like a fixed color object, a sample image data object, or a color text object] changing a tone playback curve to a different object type. Thus, a calibration print is measured, and object optimization rendering accommodation is made at step S110, and control returns to step S70 for printing a actual document.

[0107] However, if this print is not a calibration print at step S90, although control progresses to step S110 and it is not real time, as for a creator, that printed document will determine whether to be O.K. or not there. When the printed document is not O.K., control progresses to step S120, edit of the object in the print data based on a creator is attained, and control returns to step S30. In step S30, an object may be adjusted through the operator interface of the object optimization ESS 100 in advance of smoothing (it makes an object type lose generally). Since it is not necessary to return to the original non-decomposing PDL document file and an operator or a creator can perform color space conversion (an unit or plurality), tone playback curve (an unit or plurality), and/or slight accommodation in other factors, the great portion of processing required in order to prepare print data using object optimization compression and a rendering does not need to be repeated. furthermore -- since an object type is held at this point -- these types of correction -- each -- ** -- it is not repeat-like and is once made by the scan line to an object.

[0108] This edit step S120 that uses the information which can be used by the object optimization ESS 100 When processing change of the rendering relation to a document, a creator The step which calls again the document of the whole needed even when it is required only for the step which returns to a workstation, and few portions to change to a workstation display, The step which changes a document, the step which generates the new PDL version of a document, Before seeing the step which resends a document to a printer, and a new print, it differs from the Prior art in that the step which the time amount of the step which waits for a document to pass decomposition processing requires is avoidable. Before only the substep of the last of the preparation print data-processing step S30 can see a print with a new creator by the ability of rendering modification to each object on a page like modification to mid tone cyanogen separation of the specific pictorial object on a page inputting into a printer instead of, it is necessary to perform again within the object optimization ESS 100. Since the last stage of document generation is often advanced repeat processing, the amount of save of time amount is quite large.

[0109] Furthermore, the rendering control which can use a document creator by workstation Since the color on which the direct control which can often be offered by the object optimization ESS 100 differs, and it gets down, and a color printer prints a workstation display indicates that a color differs A document creator is not by editing by workstation of a Prior art, and converging quickly with the desirable color to the object in a document is expected by using the high-speed reprint step S120 accompanied by edit.

[0110] however, a certain modification whose print is O.K. and which control progressed to step S130 from step S110, and was made at step S120 there when becoming -- the object optimization ESS 110 -- or it is saved to a standard format by document generated software for

automatic inclusion on a document. Next, control progresses to step S140 and processing ends it there.
[0111] Drawing 10 shows the processing for preparing print data using object optimization compression and the rendering of step S30 of drawing 9 more to details. As shown in drawing 10 , print data-preparation processing of step S30 is started at step S200. In step S200, the next page (or the 1st) of the document generated at step S20 is obtained from the internal PDL file source means 110 or other remote PDL file source means 12 as a current page.
[0112] Next, in step S210, the object list of a current page is constituted and the object optimization rendering tag related in the specific object to this page is incorporated. In step S210, an object optimization rendering tag can be automatically generated from the object type determined to each object by the PDL decomposition means 130 and/or IOT data, and the command instruction generating means 140. Or while the automatic processing for generating an object optimization rendering tag is used as a default mode, when preparing a document at step S20, a document creator can contain clearly the object optimization rendering hint when specifying an object using PDL. These creator insertion rendering hints Defining an object type clearly, color space conversion, or a tone playback curve is defined clearly, Setting halftone screen frequency and/or an angle clearly and when that is not right The maximum laser power modulation which directs the priority over GAMYUTO mapping usually automatically set up by the PDL decomposition means 130 and/or IOT data, and the command instruction generating means 140, and/or other desirable object optimization rendering parameters, namely, the thing for which a set point is set clearly -- it contains.
[0113] Since the object list containing an object optimization rendering tag is generated to the current page in step S210, in order to guarantee that they were not canceled, the monitor of a system resource like the memory which can be used in step S220 is carried out. For example, a very complicated page like the complicated clipping region which has the sweep set up with a certain angle to the direction where a page intersects perpendicularly needs many objects for a degree with the inadequate memory resource of the RAM portion 151 of memory 150 in an object list. That is, this page includes the problem of navigation compression actually.
[0114] Thus, the fall-back mode which does not use another memory for every object of each addition is offered. In this fall-back mode, the rendering of the current page may be carried out in the resolution which the rendering was carried out to the sample channel, and fell by the system resource. If step S220 determines that a resource is not enough, control will progress to step S230 and it will prepare the print data to a current page using fall-back mode. Generally, it depends for fall-back mode on generating the conventional bit map / cutting tool map used for conventional IOT170. in this case, as for a page, the print parameter for every object is optimized -- as -- although not printed, that page may be printed at least. Furthermore, since some PDL defines the print parameter to the present page by processing overflow of a resource with reference to a front page (that is, a page not being independently) by carrying out a default to the conventional bit map / cutting tool map, it is possible to perform such reference to the parameter [as / in step S230] with which the defect page was defined above.
[0115] In step S230, attainment of the page which comes out of an available resource may start the conventional PDL decomposition processing from the 1st page of a current document. All the conditions and graphical operator (operator) parameters are reset by 0. The graphical operator from the 1st page is processed only in the limitation which maintains a graphical condition to accuracy. That is, a bit map or a cutting tool map is not written in. This continues to the independent page of the last before meeting with a defect page. Although both a graphical operator and image data are processed from the point, image data is not outputted to IOT. If a defect page is reached, the processed image data will be outputted to IOT. Processing of the continuing page and the output to those IOT(s) continue to the independent page of the beginning after meeting with a defect page.
[0116] From this point, the restart of the object optimization PDL decomposition means 130, IOT data, and the command instruction generating means 140 is carried out by the 1st page of a current document, and they reset all data fields and graphical operators to 0. The object optimization ESS 100 continues object optimization processing, without generating image data to

the independent page of the beginning after the defect page was discovered. From this point to the following defect page, the PDL decomposition means 130, IOT data, and the command instruction generating means 140 operate as mentioned above. Next, this processing is repeated for every defect page until the whole document is printed.
[0117] Or in step S230, two parallel-processing operators may be initialized to a current document. The 1st processing operator may be object optimization processing. The 2nd operator may be the conventional cutting tool map / bit map processor. An object optimization processor can be continued until the 1st defect page is discovered. At this point, the 2nd processor starts actuation, without generating image data to the last independent page before a defect page is discovered. Although image data is generated from this point to a defect page, it is not outputted to IOT170. Next, to the independent defect page after first page, it is generated and a defect page and all the continuing pages are outputted to IOT by the conventional processor. If it meets with the independent page of the beginning after a defect page, an object optimization processor will start analysis of a graphical operator, without outputting print data to IOT from a defect page to the independent page of the beginning after a defect page.
[0118] From this point, an object optimization processor generates the data by which object optimization was carried out again, and outputs it to IOT170, and it is continued until it meets with the following defective page. At this event, the graphical condition of the conventional processor is updated from the 1st independent page after the 1st defective page to the independent after [the 2nd defective page] last page. Next, above-mentioned processing is repeated until the page of the last of a current document is printed.
[0119] However, if it determines that step S220 has an enough memory resource, control will continue to step S240 and a scan line and a command will be generated [then,] with the scan line base. Next, in step S250, real-time channel data is extracted and it is stored in memory 150.

[0120] It is determined whether steps S230 and S250 continue to step S260, and have a page [document / current] further to be processed there. Supposing it is, control will return to step S200. However, if there is no page processed further, control will continue to step S270 and will return control to step S40.
[0121] Drawing 11 shows the processing for joining together using object optimization decomposition and the rendering of step S70 of drawing 9 , and printing more to details. As shown in drawing 11 , the processing which step S70 combines and is printed is started at step S300. In step S300, the compressed data to the next page (the 1st) of the set (namely, page of a current document) with which the page was collated is obtained. The data with which this following page was compressed is obtained from the RAM portion 151 of memory 150 with a bus interface 1610 through a bus 114.
[0122] Moreover, the compressed data may be stored in the nonvolatile memory portion 152 of memory 150. This is useful when it desires a reprint [without the need of repeating processing for a creator preparing print data using object assignment compression and the rendering of step S30 / the copy of a document page]. In order that the compressed data may control an object rendering function, thereby, object optimization capacity is not lost including meta-bit data.
[0123] Next, in step S310, using the multi-channel combiner 1630, compression discharge is carried out and the data with which the current page was compressed is combined. Next, in step S330, in order to optimize the page printed by IOT170, the meta-bit information offered to the IOT controller 160 is used so that the object optimization rendering method may be chosen and it may choose between simultaneous input streams.
[0124] That is, the multi-channel combiner 1630 is the object base, and in order to determine parameters, such as color space conversion, a tone playback curve, halftone generator frequency and a screen angle, the maximum laser power set point, and/or other information, it can use meta-bit information. The optimal data is determined based on processing of a different type applied to data under control of a meta-bit, and is outputted to IOT170.
[0125] Moreover, they may be used by the multi-channel combiner 1630 so that other meta-bit information may be outputted to IOT170 by the multi-channel combiner 1630, while some of meta-bit information generates object optimization data. In order that these meta-bits may be

outputted to the IOT170 control subsystem of IOT170 and these meta-bits may optimize print data further, color space conversion, a tone playback curve, the maximum laser power set point and/or halftone generator frequency, and a screen angle are included.
[0126] Moreover, all the meta-bit information is transmitted to the subsystem of IOT170 with print data by the IOT controller 160. Furthermore, some modes of object optimization renderings, such as the maximum laser power set point, color conversion, and tone playback, may be applied between the processings for preparing print data using the object assignment rendering of step S30, and compression while a meta-bit controls other modes in IOT170 in the multi-channel combiner 1630.
[0127] If a current page is printed by IOT170 in object optimization form in step S330, control will progress to step S340 and it will be determined whether the page of the last of a current copy was printed there. When the last page is not printed yet, control returns to step S300. However, if the print of the page of the last of a current copy is completed, control will continue to step S350 and it will be determined whether all the pages of hope were printed there. Supposing that is not right, it will progress to step S360, and the increment only of 1 is carried out, then the number of copies [there] progresses to step S370, the pointer of a current page will be reset to the page of the beginning of a current document, and, finally control will return to step S300. However, if step S350 determines that the last page was printed, control will continue to step S380 and will return control to step S90.
[0128] Drawing 12 shows the processing for constituting the object list which has the object optimization rendering tag of step S210 of drawing 10 more to details. As shown in drawing 12 , the configuration of an object list is started by reading a PDL document file in step S400. Next, in step S410, the language construct of the following (beginning) is obtained as a current language construct, and is analyzed. After a current language construct is analyzed, a standard graphics interface is called. The analysis of PDL led to a standard graphics interface and other actuation changes from one PDL to other PDL. These processings are well-known.
[0129] Next, it sets to step S420, and the present language construct is analyzed in order that it may determine whether direct the present end-of-page condition. If a current language construct denotes a current end-of-page condition, control will progress to step S430 and the clipper field of a current page will be verified. Clipper verification processing is further stated to details in relation to step S1140-S1160 of drawing 19 . If a current clipper field is verified in step S430, control will progress to step S440 and control will be returned to step S220.
[0130] If a current language construct does not have a finger example of a end-of-page condition current at step S420, control will progress to step S450. In step S450, a current language construct is checked in order to know whether it is a color operator. Supposing it meets, control will progress to step S460 and color operator processing will be performed.
[0131] However, in step S450, if it is determined that a current language construct is not a color operator, control progresses to step S470, and a current language construct will be checked in order to know whether it is a masking operator. If it becomes so, control will progress to step S480 and masking operator processing will be performed.
[0132] However, in step S470, if a current language construct is not a masking operator, control progresses to step S490, and the following language construct will be checked in order to know whether it is a condition operator. If it becomes so, control will progress to step S500 and graphical condition operator processing will be performed. All steps S460, S489, and S500 return to step S410.
[0133] If it is finally determined that a current language construct is not a condition operator in step S490, control progresses to step S510, and error indication will be outputted by the object optimization ESS 100 in order to direct that current language must have been analyzed appropriately. Next, control returns from step S510 to step S20 through step S520.
[0134] It should be understood that the color operator checked at step S450 contains "setcolor", "image", "colorimage", and a operator like the command of equivalence. Of course, it depends for a actual command on PDL language. Similarly, the masking operator command checked at step S470 contains "fill", "stroke", "character", and such other commands. Moreover, it depends for a actual command on PDL language. Finally, the condition operator

command checked at step S490 contains "setclipper", "setstrokewidth", "setrenderhint", and a command like such other operators. Moreover, it should be understood that it is dependent on the PDL language with which a actual command is used.
[0135] Drawing 13 shows the processing for generating the scan line data of step S240 of drawing 10 more to details. As shown in drawing 13 , it starts by initializing an active object list at the processing step S600 for generating scan line data, and setting a scan line to 1.
[0136] Next, in step S610, it merges to the active object list with which each new shelf object from a current scan line aligned. Drawing 28 shows the general-purpose form of the scan line object list data generated during the configuration of the object list which has an object assignment tag in step S210. As shown in drawing 28 , the pointer which points out the object started on the 3rd scan line 1513 points out the 1st object (beginning) 15131 on the 3rd scan line 1513, and, on the other hand, it points out the 2nd object 15132 on a scan line 1513. In this structure, an object list as shown in "15131" and "15132" contains the object positioned by the list according to the relative ranking of the corresponding PDL language construct which generates them.
[0137] however, the active (it sorted) object which aligned -- the starting position of an object -- and it aligns in right sequence from the left. Thus, the 1st object of the active object list which aligned is an active object which has a leftmost starting position. Similarly, the object of the last on each scan line is an object which has ** with a rightmost starting position. Scan line ****** each of that object is merged to an active list by processing of a given scan line so that sequencing to the right from this left may be maintained. Generally, an active object list starts in the scan line before a current scan line, and contains the object which is still active.
[0138] If it merges into the active object list with which a new object aligned at step S610, control will progress to step S620. In step S620, it is generated from the active object list with which the run list to a current scan line aligned. next, determining which run of a run list has a run list to a there current scan line in a "crowning" along with each point of a scan line by control progressing to step S630, i.e., the point, -- all -- others -- it graduates by determining the run which is not in the bottom of a run (flattened).
[0139] In step S630, smoothing of a run list advances control to step S640. In step S640, the sequencing list of the commands and colors to a current scan line is generated. A command is generated in order to ensure being set up appropriately so that that a suitable color can use with a color register, DISUIBURU [mask data / enable or] appropriately, the sampled image data being able to use if needed, and a meta-bit may optimize hardware processing of the laser modulation set point performed by compression discharge, color space conversion, tone playback curve processing, the IOT controller 160, and IOT170. The color in the form of referring to the pallet is generated in order to ensure that reading appearance of an exact color and the rendering tag information is carried out from the pallet to each instruction.
[0140] Control progresses to step S650 after step S640, and it is removed from the active object list with which the item consumed there aligned. An object is consumed when a current scan line is a scan line of the last in which an object appears. Next, in step S660, the number of scan lines is checked in order to determine whether the current number of scan lines is the scan line of the last of a page. Supposing that is not right, control will progress to step S670 and the increment of the number of scan lines will be carried out only for 1 there. Flows of control return from step S670 to step S610. However, if a current scan line is the last scan line, control will be returned to step S250 through step S690.
[0141] After scan line data is generated at step S240, control progresses to step S250. Drawing 14 shows step S250 more to details. Processing is started at step S700 by extracting, compressing and storing command data. the feature of command data sake -- Lempel - a jib -- it is used in order that well-known compression technology like - Welch (LZW) may compress command data. Command data is extracted from a smoothing run with the scan line base. Command data is stored in a part for the command channel data division of the RAM portion 151 as shown in drawing 31 .
[0142] Next, in step S710, fixed color data are extracted, compressed and stored so that it may be explained in full detail below. Next, it sets to step S720, and meta-bit data are extracted,

compressed and stored. It sets to step S730, a mask pointer is extracted and stored, and the mask data which those mask pointers point out are extracted, compressed and stored. Like [in the case of command data], in order that a Prior art like LZW may compress a fixed color and meta-bit data, it is used. Like [in the case of an above-mentioned step], fixed color-data and meta-bit data and a mask pointer are extracted from a smoothing run list, are the scan line base and are stored in the fixed color channel portion of the RAM portion 151, a meta-bit channel portion, and a mask-data channel portion, respectively. These mask data are extracted similarly and then are compressed using known 1-bit compression technology like run length encoding, CCITTGroup4, and other compatible systems. Once it is compressed, mask data are stored in the mask-data channel portion of the RAM portion 151 again.

[0143] Next, in step S740, it is stored and an extract and the sampled image data are stored in the sample data channel portions of an extract, compression, and the RAM portion 151 for the sampled image pointer. Step S710- Processing of S740 is dramatically similar. However, Joint of International Organization for Standardization (InternationalStandards Organization) Photographic Expert Although Group (JPEG) defined, different compression technology for every type [like] of data is used. Step S710- Once all parts for the data division which are different by S740, and types are extracted, compressed and (probably) stored, control will return to step S260 through step S750.

[0144] Drawing 15 shows how to process the color operator of step S460 of drawing 12 , more to details. Color operator processing is started by determining whether offer the image with which the color operator was sampled at step S800. If it is not the image with which it was sampled, control will progress to step S810. In step S810, a color palette is checked, in order that the fixed color directed with a current language construct may determine whether it already has the same entry as a color palette. If a pallet entry is established first, the rendering tag field will be set to a default. An answer is affirmation, when it checks in order to know whether there is any same pallet entry as long as the rendering tag leaves a default. If it becomes so, control will progress to step S820 and the pallet entry which has the same color model as being directed with the current following language construct there and pixel data will be detected.

[0145] However, for example, if the rendering tag is changed next, an answer is negation and a new pallet entry needs to be generated by step S1050 of drawing 16 . Thus, control progresses that the answer of step S810 is negation to step S830, a new pallet entry uses the color model and pixel data which are directed with a current language construct by ****, and it is established in a color palette. Next, in step S840, in order that a hashing function (function) may determine a suitable pallet index, it is applied to the pixel data directed with a current language construct. As shown in drawing 30 , a pallet consists of two or more 1521 to 152 n index slots. Each hash table (table) of a pallet 1520 has many pallet entries, and each of a pallet entry may be in any of a fixed color entry and the sampled entry. A pallet is stated more to details later. Next, in step S850, a current pallet entry is inserted in a pallet by the hash table slot determined by the pallet index.

[0146] Instead of the setup of a new pallet entry, a current fixed color is independently storable until it needs for step S1050 to set a rendering tag to this current fixed color. In this case, it is formed by the rendering tag field by which a new pallet entry is already reset from the default only at this event.

[0147] However, step's S's800 decision of that the color operator offered the sample image advances control to step S860. At step S860, an image resolution divisor is calculated based on image data and the IOT resolution which can be used. As stated with reference to the bits 6 and 7 of a table 4, since the resolution which can be obtained to IOT is generated, an image resolution divisor directs what clock cycle ****** of each pixel of the sampled image is carried out.

[0148] next, the latest "the current conversion (currenttransform)" evaluated by the image resolution divisor and graphical condition operator processing in which current image data was determined at step S860, in step S870 -- following -- a revolution -- and/or, a scaling is carried out. Next, in step S880, a new color palette entry uses the image data by which a current color model and current were sampled, and is established.

[0149] Next, in step S890, a pallet index is calculated by applying a hashing function to image data height (S size), i.e., "S-Size." Next, in step S900, a current pallet entry is inserted in a color palette by the hashing table slot determined by the pallet index.
[0150] Next, from either of steps S820, S850, and S900, control progresses to step S910 and the "currentcolor (current color)" pointer generated in graphical condition operator processing there is set to the current pallet entry determined by steps S820 and S850 or S900. Next, control progresses to step S410 through step S920.
[0151] Drawing 16 shows how to process the masking operator of step S480 of drawing 12 , more to details. Masking operator processing is started at step S1000 by determining whether parameter"renderhint(rendering hint)" is already set by graphical condition operator processing. Supposing that is not right, control will progress to step S1010 and an object optimization rendering tag will be determined automatically there. However, if parameter "renderhint" is set, control will progress to step S1020 and an object optimization rendering tag will be drawn from a "renderhint" parameter (an unit or plurality) there.
[0152] That is, it is determined by analyzing an object type first to a current language construct depending on the number of the level of distinction offered between the types with which the object type as which, as for the object optimization rendering tag, the current language construct was determined when becoming and object to which parameter "renderhint" is not set differ from each other. If only the single level of distinction is enabling, the object optimization rendering tag determined in step S1010 will distinguish an image object (for example, halftone object) from for example, a non-image object (for example, a text and a line art object). Supposing the level of an addition of distinction is enabling, an additional rendering tag will be generated so that saturation level which is different in the same object, which color-space-changes and is different and which tone-playback-curves and is different may be offered. It is generated for every object in order to direct the laser power set point used when at least one rendering tag writes in the data of the object in any case.
[0153] A set of a "renderhint" parameter overrides the value determined automatically, otherwise, the value determined at step S1010 with an explicit instruction from a document creator in step S1020. Thus, if a document creator is not so, it can override the rendering tag set at step S1010. Therefore, although a "renderhint" parameter may not be required, it can specify what an object type is. It may be the independent type of the hint which directs the object with which a document creator wants to gain an observer's eyes irrespective of which object type this object type is. Or "renderhint" has this object in a background and you may direct not to want to gain an observer's eyes. "renderhint" may direct the thing for which the edge of an object should be made into Sharp and which do not come out. "renderhint" may direct that saturation should not be boosted and the color by which this object was defined should be held. "renderhint" may direct that a different specific laser power set point from a default set point to the type of an object should be used instead.
[0154] Moreover, "renderhint" may define an object type by the remaining portion of analysis following a default mode clearly based on the subparameter of the defined object type and the undefined.
[0155] Next, from step S1010 and step S1020, control progresses to step S1030 and the rendering tag of the "currentcolor" condition set by graphical condition operator processing by **** is updated. If a "currentcolor" tag is not in agreement with the rendering tag to the pallet entry as which it was instructed by "currentcolor" instead of a default tag as shown in drawing 12 , and 15 and 16, the pallet entry must be copied, the rendering tag of the new pallet entry will be updated, and a "currentcolor" pointer will be updated by the new pallet entry.
[0156] Moreover, when the new pallet entry is not generated between the processing color operator step S460, color data are not inserted in a pallet and held separately. In this case, a new pallet entry is generated at step S1030, and that color value is set to the place held separately. This new rendering tag of a pallet entry is set to the value determined in step S1010 or step S1020. This new pallet entry is inputted into a pallet at this time by performing steps S840, S850, and S910 as a portion of step S1050.
[0157] Next, in step S1040, since one or more primitive (Lord) objects, such as a box and a bit

map, are generated, scanning conversion of the current object is carried out. Scanning conversion is well-known processing. Next, it is determined at step S1050 whether all the primitive objects were processed. If one of primitive objects remains, control will progress to step S1060 from step S1050, and the following primitive object (beginning) will be obtained from a masking operator as a current primitive object by ****. Next, in step S1070, primitive masking object processing is performed to a current primitive object. Next, control returns to step S1050. This loop continues until it determines that there is no primitive object by which step S1050 should be processed. In this case, control returns to step S410 through step S1080.

[0158] The more detailed version of the graphical condition operator processing stated at step S500 of drawing 12 is shown. In drawing 17 , graphical condition operator processing of step S500 is started by determining whether the "setclipper" operator was set at step S1100.

[0159] Control will progress to step S1110 and that the "setclipper" operator is not set will determine whether the "setrenderhint" operator was set to, if it becomes. Supposing that is not right, it progresses to step S1120, and control will set a graphical condition to this condition operator, and will progress to step S1180. However, if a "setrenderhint" operator is set at step S1110, control will progress to step S1130 and the "renderhint" parameter in a graphical condition will be set to the rendering hint directed by the "setrenderhint" operator there. Control progresses step S1180 again from step S1130.

[0160] However, in step S1100, if a "setclipper" operator is set, control will progress to step S1140, a there current clipper object will exist, and it will be determined whether the integrity attribute is "perfect." If it becomes so, control will progress to step S1150 and a current perfect clipper object will be changed into a sweep object there. Next, the sweep outline shown as a field r1 is set to a current clipping region. Simultaneously, the fill (restoration) field of a sweep shown in a field r2 is set to the sweep object under a current clipping region. However, when a current clipper object does not exist, or when the current existing clipper object does not have the integrity attribute of "complete (it is perfect)", control progresses to step S1160 directly from step S1140.

[0161] In step S1160, this object is inserted in a scan line alignment active object list as a scan line of the beginning of an object. Step S1160 forms the "clipper verification" processing S430 which was shown in drawing 12 and mentioned above from step S1140. Next, control progresses to step S1170 and the "currentclipper" command of a graphical field is set [then,] to a new clipper field. Next, in step S1120 and step S1130, control progresses to step S1180 and control returns to step S410.

[0162] As step S620 of drawing 13 was described, more detailed description of the processing for generating a scanning line list is shown in drawing 18 . The processing for generating a scanning line list is started in step S1200 by setting variable "thisobject" to the object of the beginning of the active object list of a current scan line.

[0163] Next, in step S1210, variable "thisobject" is tested in order to determine whether it points out an effective object. Supposing variable "thisobject" does not point out an effective object, control will return to step S630 through step S1280. However, if "thisobject" points out an effective object, it will always become truth immediately after setting "thisobject" to the first object, and control will progress to step S1220.

[0164] In step S1220, "thisobject" is checked in order to know whether it is a sweep type object. If it becomes so, control will progress to step S1230. It merges into the run list with which one run aligned for every piece of s1 outline of "thisobject" on the present scan line in step S1230. Each of the merged run consists of a starting position in alignment with the scan line, and a termination location. Each run consists of a pointer which points out the lower layer sweep object s2 of "thisobject" again, and a layer and color data may be extracted. As mentioned above, s1 field was effectively set to the clipping region, when a sweep was generated, and another side s2 field was set to the lower layer sweep object of a current clipping region. Control progresses to step S1270, after performing step S1230.

[0165] In step S1220, when the class of "thisobject" is not a sweep, control progresses to step S1240, and "thisobject" is checked there in order to know whether it is a clipper type object. If it becomes so, control will progress to step S1250. In step S1250, from a "thisobject" inside list,

each object resists the clipper of "thisobject" and clips. Inside lists are lists of objects added to this list while being collected for this clipping object between the primitive masking object processings of step S1740-S1770 which are shown in drawing 23 and stated more to details below. That is, each inside list of objects relates to the clipping object which corresponds while the clipping object which corresponds as a current clipping region was effective. Like what was mentioned above about s1 sweep outline, the clipper of "thisobject" was set to the effective clipping region, when [this] clipper object generation was carried out. An object is clipped by removing all the portions of the object on the outside of the clipper of a clipping region.
[0166] after an object clips, each profit **** run is merged into the run list which aligned. Each run consists of a starting position in alignment with a scan line, and a termination location in step S1220. However, each run consists of a layer from the inside list of "thisobject", and a pointer to the clipped object to color data in step S1250. Like the case in step S1230, once processing is completed, control will progress to step S1270.
[0167] In step S1240, when the class of "thisobject" is not a clipper, control progresses to step S1260. In step S1260, it merges to the run list with which the run for every piece of "thisobject" of the present scan line aligned. Each run consists of a starting position in alignment with a scan line, and a termination location like [in the case of step S1230 and step S1250]. However, it consists of a pointer to "thisobject" [as opposed to / **** / in this case / again / a layer and color data in each run]. Like [in the case of step S1230 and step S1250], once processing of step S1260 is completed, control will progress to step S1270.
[0168] It is set to the object variable "thisobject" is instructed to be by the "next" field of current "thisobject" in step S1270. Therefore, following "thisobject" turns into current "thisobject". Control progresses to step S1210 from step S1270.
[0169] Drawing 19 shows the processing for graduating the run list of step S360 of drawing 13 more to details. As shown in drawing 19 , the processing which graduates a run list is started at step S1300. In step S1300, the foreground and background object of a current run are initialized by the white object which has a default rendering tag. A starting position and a termination location are initialized by initiation of a current scan line. Finally, variable "currentstart" is initialized by initiation of a scan line.
[0170] Next, in step S1310, variable "currentstart" is checked in order to determine whether termination of a scan line was reached. When that is not right, control progresses to step S1320 and the next visible run is identified there.
[0171] Next, control progresses to step S1330, and it checks in order to determine whether they of the foreground and background object color, and rendering tag of the next visible run are the same as that of the foreground of the object of a current run and a background object color, and a rendering tag there. If this is truth, control will progress to step S1360 and a run will be combined there by setting the termination location of a current run to the termination location of the next visible run. However, when this is not truth, control progresses to step S1340 and the there current command and there current color to a run are generated. Next, in step S1350, the next visible run turns into a current run.
[0172] Then, both steps S1350 and S1360 progress to step S1370, and it is set in the termination location of the present run of "currentstart". Next, the control from step S1370 progresses to step S1310. In step S1310, if "currentstart" reaches to termination of a scan line, control will progress to step S1380 and will be generated there for a current run of the set of the last of a command and a color. Control returns from step S1380 to step S640 through step S1390.
[0173] Drawing 20 shows the processing for generating the command and the present present color to a run of a scan line of steps S1340 or S1380 more to details. [of drawing 19] In step S1400, the foreground object of a current run is investigated, in order that the object may determine whether to be transparence or not. That is, the enquiry is whether to be the object 15570 of the bit map type with which the object has the true clear field 15574. Supposing that is not right, control will progress to step S1410. In step S1410, the foreground object of a current run is investigated in order to know whether the object is the class sweep type object 15580. Supposing that is not right, control will progress to step S1420 and will generate the Normal

command and a color. Next, control progresses to step S1460.
[0174] If it is determined that a foreground object is the sweep class type object 15590 in step S1410, control will progress to step S1430 and a sweep command and a color will be generated. Next, control progresses to step S1460 again. However, in step S1400, control progresses to step S1440 with decision **** in case of the object of the class bit map type with which the foreground object of a current run has the true clear field, a background chain is processed, and a mask-bit map is fixed. Next, control progresses to step S1450 and generates a command and a color using a mask. Next, control progresses to step S1460 again.
[0175] This procedure is not common in the point that therefore it may be called to two steps. thus, the step S1460 -- which of steps S1340 and S1380 -- this procedure -- call appearance -- or [bottom] -- determining -- the following step S1350 with appropriate control, i.e., a step, and S1370 -- it returns. That is, if generation of a command and a color is called at step S1340 to a current run procedure, control will progress to step S1350. Similarly, if generation of a command and a color is called at step S1380 to a current run procedure, control will progress to step S1390 and control will be returned to step S640.
[0176] As mentioned above, step S1400 operates so that the object of the present run may determine a foreground color, while the object directed by that background field determines the color which is "0" bits of the mask of this run, when the foreground object of the present run is transparence (namely, bit map class object). However, you may be the transparence bit map which is the same or has a different foreground color in itself [background object]. Two adjoining clear layers may be combined by applying a logic OR function to those bit maps like [when a foreground color is the same] the case where it has a black alphabetic character bit map. Since the original writing of the mask channel map of this field was overwritten by the lower bit, OR of the bit map of a lower bit map object is carried out to that map at this step. Carrying out OR of the transparence background to the object on it, and joining together can continue until it reaches a transparence background layer using the color of different "1" bit from the upper object. At this point, the conflict in the desirable activity of a mask channel was seen. It is solved by changing into a run the group of "1" bit which adjoins spatially [the transparence background bit map which carries out a conflict], and applying "smoothing" processing recursively. Thus, the transparence background object which carries out a conflict is changed into the set of the object which is not transparent. This processing follows a background chain until an opaque object or a layer "0" white paper is discovered.
[0177] Drawing 21 shows more detailed description of the fixed color extract of step S710 of drawing 14 , compression, and storing processing. An extract, compression, and storing fixed color processing are started at step S1500. In step S1500, following referring to the color (criteria) (beginning) is obtained as current refer to the color. Present referring to the color is a pointer which points out one of the pallet entries of the pallet data structure shown in drawing 30 .
[0178] Next, in step S1510, color data are obtained from a current pallet entry. The color class field 15233 which directs whether it is the image pallet entry which the rendering tag field 15232 which offers the following link 15231 to which each fixed color type palette entry 15230 points out the next pallet entry of this hash table slot, and the data about what the rendering of this pallet entry should be carried out, and this are fixed color palette entries as shown in drawing 30 , or was sampled, or it is the pallet entry of other types is included. The color model field 15234 directs the desirable color space conversion for changing this color into CMYK with a rendering tag field while directing which type of color model (for example, RGB, CMYK, CIELab, or in addition to this) pixel data uses. Finally, the pixel value field 15235 stores actual color data.
[0179] Next, in step S1520, reading appearance of the pixel data of a current pallet entry is carried out from the pixel value data field 15235. Next, in step S1530, the pixel data by which reading appearance was carried out exactly is stored in the channel data structure of RAM151 in the location which can be used for the degree of a fixed color channel, as shown in drawing 31 .
[0180] It is foreknown that the pixel data which consists of mixture of a color model is stored in a channel data structure in this way. It is changed into CMYK (or other IOT assignment) data under control of a meta-bit in IOT170 or the multi-channel combiner 1630 as mentioned above.

Moreover, the color conversion for which the color model and the rendering tag field opted can be applied to pixel data, and can change it into CMYK as a part of step S1520.
[0181] Next, in step S1540, a pallet is checked in order to determine whether current referring to the color is last referring to the color. Supposing it is not so, control will be again chosen as step S1500 as current in return and its following refer to the color refer to the color. However, when current referring to the color is last referring to the color, control progresses to step S1550.
[0182] In step S1550, from memory, reading appearance of the color data in fixed color channel data channel structure is carried out, they are compressed, and are re-stored in compression form. The data transfer through a bus 114 is minimized by compressing the fixed color data stored in the fixed color channel field like [in the case of other channel data].
[0183] If fixed color data are compressed and it is stored in step S1550, control will return to step S720 through step S1560.
[0184] Drawing 22 shows more detailed description of the meta-bit extract of step S720 of drawing 14 , compression, and storing processing. As shown in drawing 22 , the processing which extracts, compresses and stores a meta-bit is started at step S1600.
[0185] In step S1600, following referring to the color (it being a pointer to a pallet entry) (beginning) is obtained as present refer to the color similarly in step S1500 of drawing 21. Similarly, in step S1610, a rendering tag is obtained from the rendering tag field 15232 of the fixed color palette entry 15230, and the sampled rendering tag field 15242 of the image pallet entry 15240 similarly in step S1510. Since both both the fixed color palette entries 15230 and sampled image pallet entries 15240 contain a rendering tag, this processing is not limited to a fixed color palette entry as in the flow chart shown in drawing 21 .
[0186] If a rendering tag is obtained at step S1610, control will progress to step S1620. In step S1620, the lookup of the rendering tag is carried out in the translation table which changes assignment between a printer independent rendering tag and an IOT assignment meta-bit. Thus, lookup processing of step S1620 returns the IOT assignment meta-bit value which offers suitable hardware **/suitable for IOT170 of assignment, or software processing control based on the rendering tag of a current pallet entry.
[0187] When a meta-bit value is acquired at step S1620, control progresses to step S1630 and it is stored in the location which can be used for the degree of the meta-bit channel of a channel data structure by which the meta-bit value over the there present pallet entry is shown to drawing 31 .
[0188] Next, in step S1640, current referring to the color is checked in order to know whether it is last referring to the color. If that is not right, control will be again chosen [to step S1600] as current in following refer to the color refer to the color return and there. However, when current referring to the color is last referring to the color, control progresses to step S1650.
[0189] In step S1650, the meta-bit data stored in the meta-bit channel are compressed, and it is stored in the meta-bit channel of a channel data structure. Next, control returns to step S730 through step S1660.
[0190] Drawing 23 shows more detailed description of primitive masking object processing of step S1070 of drawing 16 . As shown in drawing 23 , primitive masking object processing is started at step S1700. In step S1700, a current primitive masking object is checked in order to determine whether it is a bit map primitive. As mentioned above, primitive objects are a box, a bit map, etc. If it becomes so, control will progress to step S1710, it will act to a mask-bit map as the Brit of this bit map primitive there (Blitted), and this bit map primitive will overwrite the data beforehand stored in the location which acts as a Brit. "A BURITTI ring (Blitting)" is a Prior art which is called "bit level block transfer" or "BITBLT" processing, and enables modification of memory block on a cutting tool or not the word boundary but a bit boundary. Next, control progresses to step S1720 from step S1710. When a current primitive masking object is not a bit map primitive, control progresses to step S1720 from step S1700.
[0191] Next, in step S1720, a current primitive masking object is checked in order to determine whether a clipping object more restrictive than a page boundary box is effective. When such a clipping region is not active, control progresses to step S1730 and this object is added to the

scan line object list corresponding to the first scan line to this object there. That is, this object is added only to the scan line object list of a scan line with which it appears first. Next, control progresses to step S1780 from step S1730.

[0192] However, when clipping regions other than a page boundary box are effective, control progresses to step S1740 from step S1720. In step S1740, a current primitive masking object is checked in order to determine whether be the portion of an imperfect sweep, although it is existing. If it becomes so, control will progress to step S1750 from step S1740. In step S1750, a current primitive masking object is added to the sweep subitem of a current clipper object. Although it is existing, one imperfect type of a sweep is a simple primitive object like a box. When a current primitive masking object adjoins the existing primitive object and is discovered, a new sweep object is generated and a primitive object current by existing is linked to the s2 field of a sweep. This new imperfect sweep object becomes the subitem of a clipper. Next, control progresses to step S1760 and the integrity attribute of a current clipper object is updated there. The integrity attribute of a current clipper object directs whether sufficient primitive masking object was added to the sweep subitem corresponding to a current clipper object, and was thoroughly filled up with the current clipper object (fill). From step S1760, control progresses to step S1780 again.

[0193] A current primitive masking object is existing, or if it is not the portion of an imperfect sweep, it will progress to step S1770 from step S1740, and a current primitive masking object will be added to the item list of current clipper objects there. Control progresses to step S1780 from step S1770. In step S1780, control is returned to step S1050.

[0194] As for drawing 24 , more detailed explanation of step S1320 of drawing 19 of the next visible run discernment processing is shown. As shown in drawing 24 , the next visible run discernment processing is started at step S1800. In step S1800, variable "thisrun" is initialized to the next run which remains in the run list which aligned. Variable "currentend" is set in the end of the next run.

[0195] Control progresses to step S1810 from step S1800. In step S1810, the run list which aligned is checked in order that it may know whether it is empty, and variable "thisrun" is checked in order to know whether it will begin after variable "currentend". If these both are falses, the run which control progresses to step S1820 and is referred to by "thisrun" there will be checked in order that the object referred to by it may know whether it has a layer on the layer of the run segment directed by variable "highestrun". That is, "highestrun" points out a run segment, i.e., the portion of a run, and it has a list of a starting position, a termination location, a foreground object, and background objects chain-ized potentially. When the layer of the object referred to by "thisrun" is not on the foreground layer of the run segment referred to by "highestrun" next, the object referred to by "thisrun" is directly under the foreground of the run segment referred to by "highestrun". In this case, control progresses to step S1830 and the object relevant to "thisrun" and it is processed as a new run [directly under] there. Control progresses to step S1860 from step S1830.

[0196] However, if the layer of the object referred to by "thisrun" is on the foreground layer of "highestrun", control will progress to step S1840. In step S1840, the starting position to "highestrun" is checked, in order that it may know whether it will start behind the starting position directed by "currentstart".

[0197] When this is not truth, control progresses to step S1850, run "thisrun" is processed there, and it becomes the new highest run. Control progresses to step S1860 from step S1850. In step S1860, it is obtained from the run list with which new "thisrun" aligned. Next, control returns from step S1860 to step S1810. However, in step S1840, if "thisrun" begins after "currentstart", control will progress to step S1870 from step S1840. In step S1870, it is set so that the termination location directed by variable "currentend" may become equal to the starting position of "thisrun". Next, control progresses to step S1880 from step S1870.

[0198] Similarly, in step S1810, if it is truth any of a test they are, control will progress to step S1880. The identified run is processed in step S1880. Next, control returns from step S1880 to step S1330 through step S1890.

[0199] There is a list of active runs on a current scan line at the beginning of smooth run list

(Flatten Run List) processing. This list aligns from the left to the right based on those extreme left points. Smooth run list processing generates from the left the stream of the command which states each portion or segment of each visible run on a scan line to accuracy in right order. The very important portion of this processing is starting at the left end of a paper, and, identifying the next longest portion of a visible run on the whole. This is attained by the visible segment discernment (Identify Next VisiBle Segment) processing next to step S1320. The fundamental approach is simple. Since a run aligns with those starting positions, they can be added to an "active run list" and the list of runs is considered to be active with this point of meeting a scan line now. Since a run aligns with those starting positions along with a scan line, they can be added to an active run list, when it arrives at those starting positions, and when the location in alignment with a scan line moves forward across those termination locations, they may be canceled. Next, the run which has the highest target value is identified as a "crowning (on-top). However, two key factors, an engine-performance factor, and a transparence bit map object complicate processing.

[0200] Some graphical structures are conspicuous in PDL which makes processing of this simple type difficult. An example is known as a radial sweep. A radial sweep consists of an object with the big lowest layer, and a gradual more small object on a lower layer more, and the object of all tops enters completely substantially inside [all] the object of such the bottom. a page top -- each -- such a sweep can have hundreds of layers. One problem has very many objects seen during an active run list near the object of a stack a crowning or near it. The gestalt of implementation of instantiation expressed here is mitigated to abbreviation completeness by introducing directly under chain-ized processing of step S1830 of drawing 24 . Directly under chain-ization removes the present highest layer from an active run list by linking to the run which covers it, as long as the run of a lower layer re-appears after the present higher run is completed when it is covered by the higher layer. When a low run is completed by higher run before, a lower run may be thoroughly canceled in that it is covered. Thus, since the object covered temporarily is chain-ized instead of being allotted to an active run list, an active run list is maintained very short. Thus, all runs of a radial sweep are in the long chain directly under a current top run. After the upper run is completed more, the run of the beginning of the chain [directly under] of the run is added to an active run list. If a new run encounters the run list which is not in a crowning and which aligned at the point, it will be added to the chain directly under a suitable layer. That is, it is added under the run which is the upper part of the run which has a layer below self, and has a higher layer. Furthermore, a run is canceled as soon as ending before being directly under [where it was removed from the chain / directly under /, and they were newly inserted] a run and making insertion is known. It means that it does not become visible [those runs] already [this] at the existing point to the right.

[0201] The 2nd complexity is caused by the transparent bit map object. "0" bits are transparence (clear), although a bit map object has the special feature of transparence and "1" bit is colored there. That is, the color of the object [directly under] of a bit map object is transparent, and visible. These both complicate chain-ization [directly under] and do the conflict of the activity of a mask channel. The latter difficulty is processed with the technology later mentioned as a portion of the discernment run processing steps S1350 and S1380 of drawing 19 . If a directly under chain is always emptied, it returns to an active run list and all the transparent runs are removed from an active run list when the difficulty of chain-izing has a transparent top run, in the gestalt of implementation of instantiation, it will be processed by reconstructing a chain [directly under] again.

[0202] Drawing 29 shows the expression with which the general-purpose object data structure 1550 stored in memory 150 was used widely. The general-purpose object structure 1550 consists of these numbers and types that change in a link field 15510, the layer field 15520, the pallet entry pointer field 15530, the object class assignment procedure field 15540 (the above-mentioned field is a fixed field), the following object data field 15550, and a following object class. Specifically, the next link field 15510 is used in order to form the list of objects by pointing out the link field 15510 of other one object as follows. Of this mechanism, the scan line object list 1510 shown in drawing 28 is formed.

[0203] The layer field 15520 is used in order to encode the relative height of an object along with the Z-axis prolonged vertically from a page. That is, the PDL file which describes a page image is constituted so that it may be covered by those objects defined after the graphical object of the page image previously defined as a PDL file filing. This object defined previously can be stated to the stack of the object which forms a page image that it is more low along with the Z-axis. It can be said that similarly the object which appears in a PDL file later is more high along with the Z-axis. The layer field 15520 encodes this relative height of the object of a stack. Each new shelf object can give the level value which becomes high continuously during primitive masking object step S1070 processing of drawing 16 .
[0204] The pallet entry pointer field 15530 is a pointer to the entry in the pallet data structure 1520 of memory 150. The pallet entry referred to may be a pallet entry generated as a result of the color operator processing step S460 which is restricted neither to fixed color data nor the sampled image data, but is shown in drawing 15 . The object class assignment procedure field 15540 is a pointer to the collection object of a procedure which changes from one object class to other object classes in detailed actuation of a procedure. Thus, the object class assignment procedure field 15540 of all the objects that have the same type or the same class shows the same procedure. The procedure of an assignment object class can access accuracy to the data of the object class assignment.
[0205] The object assignment class data field to the box list class data field 15560 shown also in drawing 29 consists of the linked box pointer field 15561 and the present box pointer field 15562. Both the linked box pointer fields 15561 point out a series of boxes which form an object. Each box of the box of this ****** consists of the link to the next box of this box of a series of, the lower left of this box, and the location of an upper right corner. Some well-known technology allots useful constraint conventionally to the box which may be expressed by such a series of boxes. Some constraint is useful although the engine performance of the clipping procedure which acts on the clipping region described by such box that stands in a row is increased. Thus, the current box pointer field 15562 of the box list class 15562 is offered as a thing convenient for clipping and other procedures.
[0206] In the bit map assignment class data field 15570, this bit map has boundary box data of that self including the bit map object pointer field 15571. The data bit which has the value of 1 in a bit map directs to say that the object in the point expressed with the bit should be printed in the color referred to by the pallet entry pointer 15530. The data bit which has the value of 0 takes one side of two semantics which carries out alternation depending on the value of the clear field 15574. In the case of 0 to which the clear field 15574 expresses a false, the data bit of 0 of a map expresses white. the clear field 15574 expresses truth -- un--- when it is 0, the data bit of 0 of a map has a transparent bit map at those points, and means that a color is determined by the object under a bit map object.
[0207] The outline object pointer field 15572 expresses the outline or the boundary of the bit map object 15570. Generally the outline object referred to is a box list class object. Thus, a complicated configuration can be expressed and bit map 15770 the very thing is able to be a rectangle. When a bit map is transparent, the background object pointer field 15573 is used between data smoothing, so that it may be directed by the clear field 15574.
[0208] The clip assignment class object data field 15580 includes the clipper object pointer 15581, the object inside clipper field pointer 15582, and the integrity attribute 15583. The clipper object pointer 15581 points out a box list class object, in order to specify the configuration of a clipper field so that it may be set by the "setclip" operator step S1170 of drawing 17 . When the clipper field expressed by this clip class object is the present clipping region, the object inside pointer 15582 is processed at the processing primitive masking object step S1770, as shown in drawing 23 . The integrity attribute field 15583 is used in order to encode by what kind of method perfect sweeps should be collected thoroughly how selectively, while the current clip class object showing a clipping region is filled up with the boundary box of a clipping region (fill).
[0209] The sweep assignment object class data field 15590 consists of s1 object pointer field 15591, s2 object pointer field 15592, the sweep change rate field 15593, and the coloring run pro juicer method field 15594. In order that s1 object pointer field 15591 may point out the outline

object of a sweep, it is mainly used, and a box list class object expresses an effective clipping region, when the objects which consist of a sweep are collected. s2 object pointer field 15592 points out the 2nd object of a class sweep, and the s1 and s2 object points out two edges of the linked list of the simple object showing the colored slice from which a sweep changes. The change rate field 15593 of a sweep is calculated, when the clip class object used in order to collect objects on the inside of a clipping region is changed into a sweep class object as are shown in drawing 17 , and shown in the graphical condition operator processing step S1150. It is used in order to determine whether be the color from which a sweep changes by sufficient frequency to guarantee the activity in the "slow sampling channel" mode above-mentioned actuation.

[0210] The coloring (color) run pro juicer method 15594 is the procedure of the type proper of the collected sweep. The sweep which changes vertically, the sweep which changes horizontally, and the sweep which has other various features are different color run pro juicer method procedures. This procedure is called between the generation sweep command shown in drawing 20 , and the color step S1430, and generates a command and a color for each [which this field 15594 can point out] coloring slice of every.

[0211] Drawing 25 shows the Normal command of step S1420 of drawing 20 , and more detailed explanation of color generation processing. The Normal command and color generation processing are started at step S1900, and the color class of an object is checked there in order to determine whether have the image with which the object was sampled as the color. If it is no, control will progress to step S1910 and it will secure that the suitable color data and the meta-bit value over this run are loaded to one of the color registers at this step. It is determined whether it was compared with refer to [by which referring to the pallet to the foreground of this run is held at each of a shadow register to a color / meta-bit register 0-2] the pallet, and the color referred to by the object was loaded to one of the color registers before. When this is not so, color data are loaded by the command which one of a color / the meta-bit registers 0-2 is chosen, and is generated at the following steps. This selection can use all conventional processings like the "minimum activity frequency (least recently used)" processing or other same or equivalent processings. The reference to the pallet corresponding to this run is outputted to the last in the location which can be used for the degree of a fixed color channel data structure.

[0212] Next, control progresses to step S1920 and the Normal command which has a suitable bit value over the specific color register chosen by the color register-select value A, a load color, and this color register-select value A is taken out there based on the value determined at step S1910. Furthermore, if needed, many "repeat" commands are generated in order to extend a run exceeding the 64-pixel length limit in the Normal command. From step S1920, control progresses to step S1960 and control is returned to step S1460 there.

[0213] When it is the image with which the color of an object was sampled in step S1900, control progresses to step S1930, and thereby, the meta-bit value corresponding to a sample image ensures ** loaded to the meta-bit register 3, when an image pixel is displayed. A shadow register is held at the RAM portions 151 of the whole command and the memory 150 between color palette reference generating processings, and can determine the current content of the color register by it. In this case, the value of the color register 3 is not important. Only the value of the meta-bit register 3 is used and color pixel data is supplied by the sample channel. When the pre value of a meta-bit register needs to be loaded based on the content of the shadow register, the pallet entry which referring to the color palette is inserted in the degree of a fixed color channel in an available location, and has an exact rendering tag and the color value which is not not much important is referred to.

[0214] After the value of the meta-bit register 3 is verified at step S1930, control progresses to step S1940, a command displays sample channel data there, a suitable sample divisor is set up, and the meta-bit register determined at step S1930 is loaded to the location which can be used for the degree of a channel memory portion. As in step S1920, if needed, a "repeat" command is generated so that run length may be extended exceeding a 64-pixel limit of the Normal command.

[0215] Next, in step S1940, control progresses to step S1950 and the address and length of sample image data which are supplied to the command generated exactly there are loaded to the next available location of a sample pointer channel data structure. In order that the extract of step S740 of drawing 14 , storing sample image pointer processing and an extract, compression, and storing sampling image data processing may discover and extract the required portion of a sample image by the DMA controller in addition to using the address and length, the address and length are used.
[0216] Next, control returns from step S1950 to step S1460 through step S1960.
[0217] Drawing 26 shows the sweep command of step S1430 of drawing 20 , and more detailed description of color generation processing. A sweep command and color instruction generation processing start at step S2000, and the change rate of a sweep is checked there in order to know whether it is 2 pixels or less per color step. In order to generate a actual command and a actual color, the sweep color RAMPURO juicer (generation) method found by the sweep tbe data of an object data structure is called. This method changes with types of a sweep, extracts the background slice of a sweep, and offers initiation/termination location for every subrun in a pallet entry and a sweep object. For example, the sweep which changes from the scan line whose color is one only to the following scan line has the color run pro juicer method that find one suitable slice to a **** scan line, and only one ** notifies it to a command generating processor. Next, a command generating processor generates "a command and refer to the color to the whole run length." By contrast, the sweep which carries out a step from a color to a color along with a scan line must have "one command and one refer to the color" for every slice of an object along with a scan line. The change rate of a sweep will be calculated between clipper verification processings, once the sweep is changed from one clipper object. This field directs the number of averages of the pixel used for every color step in a sweep. When a change rate is 2 pixels or less per color step, control progresses to step S2020 and one single command which chooses a color / meta-bit register 3 using a color register-select B value is generated. As mentioned above, when using the color / meta-bit register 3 which has a color register-select B value, one color is read from a fixed color channel for every clock cycle. By using this slow sample channel, only a single command is required with refer to [of one **] the color palette for every color slice. A command bit and referring to the color palette are outputted to a location available to the degree of command channel memory and fixed color channel memory. [which set up an IOT controller]
[0218] Next, control progresses to step S1460 through step S2030 from step S2020.
[0219] However, if the change rate of a sweep exceeds 2 pixels, control will progress to step S2010. In step S2010, a command and referring to the color palette are generated, and it must be loaded to command channel memory and fixed color channel memory. Control progresses to step S1460 through step S2030 like [in the case of step S2020] from step S2010.
[0220] Drawing 27 shows more detailed description of the command which uses mask processing of step S1450, and color generation. In order to establish the conditions which enable a combiner to run this processing to Normal (activation), it is determined whether there is any special case where the command of length 1 must be published. Such conditions include the need for loading both a meta-bit value and a fixed color, when carrying out the switch between the foreground of an image, and a fixed color foreground using the need for loading the color of both transparence sweeps, or a mask. [in some of these cases], it must be investigated in order to determine the sequence that the bit color register of the beginning (the 1st) of the mask data itself is loaded, and the color chosen by it by the pixel of the beginning of mask data chooses the color register loaded by the command which takes out the pixel. The 2nd command is taken out, and the first command has a length of 1 pixel so that other required color / meta-bit registers can be loaded to the 2nd pixel clock. In some cases, mask data need for it to be reversed and to be processed. It is generated in order to choose a fixed color on an image using a mask. Since a sample channel is chosen by the color register-select A value, the mask selection between the SrcB fields in SrcB and a command must be specified in order to allot the fixed color which must be chosen by the color register-select B value of this example in this case to a foreground.
[0221] The processing is started in step S2100, and a foreground and a background color are

checked there in order to know whether it is the image color by which those any were sampled. In affirmation, control progresses to step S2110 and an image meta-bit and a fixed color are verified there for the mask. That is, the load sequence of a color register and a meta-bit is determined in order that the processing may ensure operating to accuracy. This decision is made by asking the initial bit of mask data by determining whether it determines whether there is any fixed color needed for any of a color register they are, or there is nothing, and the meta-bit value of the meta-bit register 3 needs to be loaded clearly. Next, since a command is generated at the continuing step, the number of the 1-pixel commands determined are needed here since hardware is initialized suitable for a suitable condition is used.

[0222] Next, control progresses to step S2120 from step S2110, and the data determined at step S2110 is generated actually.

[0223] Next, control returns from step S2120 to step S1460 through step S2190.

[0224] However, if it is not the image color by which the foreground and the background were sampled in step S2100, control progresses to step S2130, and a shadow register will be checked there in order to determine whether both a foreground and a background color need to be loaded. If it becomes so, it will be determined which should be first loaded among two colors in which the need of progressing to step S2140 and being loaded from step S2130 has control. Moreover, this is made by investigating the bit of the beginning of mask data as mentioned above. When the color which needs to be loaded first is determined at step S2140, control progresses to step S2150 and a 1-pixel length command is generated there, and since the first color is loaded, it is loaded to the location which can be used for the beginning of command channel memory. Of course, the first color is loaded to the location which can be used for the degree of a fixed color channel.

[0225] Next, control progresses to step S2170 and it is loaded by generating the command to the remainder of the run length who has the mask which the color determined are loaded to the 2nd there chooses appropriately between the 2 color. Thus, the 2nd command is loaded to the location which can be used for the degree of command channel memory, and 2nd referring to the pallet is loaded to the location which can be used for the degree of a fixed color channel.

[0226] However, when both a foreground and a background color do not need to be loaded at step S2130, control progresses to step S2160, and a shadow register is checked there in order to determine whether either a foreground or a background color needs to be loaded. If it becomes so, control will progress to step S2170. If that is not right, control progresses to step S2180, a command is generated there, and it is loaded to the location which can be used for the degree of command channel memory. However, since both colors are already loaded to the register of an IOT controller, reference of an addition on a pallet is generated and it does not need to be loaded to fixed color channel memory. Next, control progresses to step S1460 through step S2190 from step S2180.

[0227] Drawing 31 shows symbolic drawing of the RAM portion 151 of memory 150. The RAM portion 151 contains a color palette 1530, the mask-bit map 1560, a pallet 1520, the scan line pointer stack 1510, the command channel memory 153, the fixed color channel memory 154, the sample color channel 155, the mask channel 156, and the meta-bit channel 157.

[0228] Drawing 32 shows symbolic drawing of the whole object optimization system which has the IOT controller 160 integrated by one system, the decomposition system 130, and a command instruction and data generation system, and has the data flow and procedure which were able to be set in order from initiation to termination. An available object optimization resource is also shown in the IOT controller 160, IOT170, and/or the decomposition system 130.

[0229] Although it was explained with the gestalt of the specific instantiation-operation in which the utility of this invention includes the system and method of this invention as outlined upwards, a clear thing has many substitution, modification, and a variation clear to this contractor. Therefore, as mentioned above, the structure used with the system and method of this invention is instantiation, and does not mean limiting this invention. Various modification does by the pneuma of this invention, and within the limits.

[0230] For example, an image may be processed by segmenting the image to a bigger segment, i.e., a field, than the size of a pixel. It can be based on how many there is how many the set point

to the laser power used in order to expose all the specific fields of an image has a halftone object or its portion in a given field configuration, i.e., the field, a text object and a line art object, or such an object. Therefore, a different laser power set point may be used in order to expose a field based on the ratio of object of different type, i.e., halftone object, text, and/or line art **. For example, a low set point can be used with the field only containing a halftone object, a middle set point can be used with the field containing a halftone object and a text, and/or a line art object, and a high laser power set point can be used with the field only containing a text and/or a line art.

[0231] that these fields are well-known distinguishable from each other, or the type of each field which uses a segment bit based on the segment algorithm developed in the future and the space coordinates of each field -- or other **** -- or it may be identified by the method developed in the future. For example, a print system user can input data at a terminal through the interface for drawing the field which should be printed using various modes of operation based on the laser power set point used when forming these fields.

[0232] Although the structure and the device which perform a laser modulation based on the difference between a halftone field, a text, and/or a line art are not necessarily limited, they are contained in all the digital copiers that modulate the output reinforcement of the power of for example, a printer, a facsimile machine and a leather raster output scanner, an LED image bar scanner, or a luminescence device and that use well-known or the device developed in the future, for example.

[0233] This invention can be carried out with super-high-degree-of-accuracy printing which uses the method of carrying out the rendering of the different halftone image from a traditional method. Super-high-degree-of-accuracy printing is performed by carrying out the rendering of the halftone dot using a different configuration. United States Patent 5485289th given to the curry (Curry) is indicating in the details of super-high-degree-of-accuracy printing. The traditional half toning method which uses a "threshold array" may also be used.

[0234] Although the utility of this invention was finally mentioned above using the various instantiation structures which generate a print image using laser, the system and method by this invention may be used with an ink jet printer like it. Therefore, a different set point as compared with the amount of ink used when not using a laser modulation, generating a halftone field and the amount control of ink generates a text and/or a line art, in order to change the size of exposure SUPPOTO is used inevitably.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1] The block diagram of an object optimization electronic subsystem is shown.
[Drawing 2] The block diagram of the gestalt of operation of the 1st of an IOT controller is shown.
[Drawing 3] The gestalt of operation of the 1st of multi-channel KOMBANA is shown.
[Drawing 4] The gestalt of operation of the 1st of FIFO control, and a command / mask channel processor is shown.
[Drawing 5] The gestalt of the 1st desirable operation of the bus interface of an IOT controller is shown.
[Drawing 6] The gestalt of operation of the 2nd of an IOT controller is shown.
[Drawing 7] The gestalt of operation of the 3rd of an IOT controller is shown.
[Drawing 8] The gestalt of operation of the 4th of an IOT controller is shown.
[Drawing 9] Flow drawing of the whole object optimization processing method is shown.
[Drawing 10] Flow drawing for preparing print data using an object optimization rendering and compression is shown.
[Drawing 11] Flow drawing for joining together and printing using object optimization compression discharge and a rendering is shown.
[Drawing 12] Flow drawing for constituting the object list which has an object optimization rendering tag is shown.
[Drawing 13] Flow drawing for generating scan line data is shown.
[Drawing 14] Loading and flow drawing for compressing are shown for real-time data.
[Drawing 15] Flow drawing for processing a color operator is shown.
[Drawing 16] Flow drawing for processing a masking operator is shown.
[Drawing 17] Flow drawing for processing a graphical condition operator is shown.
[Drawing 18] Flow drawing for generating a scan line run list is shown.
[Drawing 19] Flow drawing for carrying out flattening of the run list is shown.
[Drawing 20] Flow drawing for generating a command and a color for a current run is shown.
[Drawing 21] Flow drawing for extracting, compressing and storing fixed color data is shown.
[Drawing 22] Flow drawing for extracting, compressing and storing metadata is shown.
[Drawing 23] Flow drawing for processing a basic masking object is shown.
[Drawing 24] Flow drawing for identifying the next visible run is shown.
[Drawing 25] Flow drawing for generating a usual command and a usual color is shown.
[Drawing 26] Flow drawing for generating a sweep command and a color is shown.
[Drawing 27] Flow drawing for generating a command and a color using mask data is shown.
[Drawing 28] Drawing of the data structure stored in memory 150 is shown.
[Drawing 29] The general-purpose structure for every object on a scan line is shown.
[Drawing 30] The general-purpose structure over a color palette is shown.
[Drawing 31] A general-purpose channel data structure is shown.
[Drawing 32] System flow drawing and resource drawing are shown.
[Drawing 33] System flow drawing and resource drawing are shown.
[Drawing 34] The block diagram of the gestalt of the 1st desirable operation of the image

processing system in IOT is shown.
[Drawing 35] The block diagram of the gestalt of operation of the 1st of object optimization print measurement and an accommodation system is shown.

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2000－153640
(P2000－153640A)
(43)公開日　平成12年6月6日(2000.6.6)

(51)Int.Cl.'	識別記号	FI		テーマコード'(参考)
B41J 2/52		B41J 3/00	A	
H04N 1/23	103	H04N 1/23	103Z	

審査請求 未請求 請求項の数3 OL （全67頁）

(21)出願番号　特願平11－327652

(22)出願日　平成11年11月18日(1999.11.18)

(31)優先権主張番号　196165
(32)優先日　平成10年11月18日(1998.11.18)
(33)優先権主張国　米国（US）

(71)出願人　590000798
ゼロックス コーポレイション
XEROX CORPORATION
アメリカ合衆国 06904－1600 コネティカット州・スタンフォード・ロング リッチ ロード・800
(72)発明者　ダグラス エヌ. カリー
アメリカ合衆国 94025 カリフォルニア州 メンロ パーク レランド アベニュー 221
(74)代理人　100079049
弁理士 中島 淳 （外1名）

(54)【発明の名称】　プリント方法及びプリンタコントローラ

(57)【要約】
【課題】　PDL等を使用して定義されたページ画像に各タイプのオブジェクトを形成するために使用されるレーザパワーが最適に制御又は選択されるようにプリントデータ及びプリンタ制御コマンドを使用するプリンタコントローラ装置及び方法を提供すること
【解決手段】　プリント方法は、異なるタイプのデータを含む画像データのバイト間を区別するステップと、第１のステップポイントでレーザパワー駆動信号を使用して第１のタイプのバイト画像データをプリントするステップと、第２のステップポイントで前記レーザパワー駆動信号を使用して第２のタイプのバイトの画像データをプリントするステップと、を備え、前記第１のステップポイントが前記第２のステップポイントと異なっている。

1

【特許請求の範囲】
【請求項１】　異なるタイプのデータを含む画像データのバイト間を区別するステップと、
第１のステップポイントでレーザパワー駆動信号を使用して第１のタイプのバイト画像データをプリントするステップと、
第２のステップポイントで前記レーザパワー駆動信号を使用して第２のタイプのバイトの画像データをプリントするステップと、を備え、
前記第１のステップポイントが前記第２のステップポイントと異なっている、プリント方法。
【請求項２】　バイトの画像データに対応するオブジェクトのタイプに基づいてそのバイトの画像データをプリントするために使用される駆動信号セットポイントを調節するプリンタコントローラであって、
プリントデータ及びメタビットデータを受信する画像出力端末を備え、前記画像出力端末が前記画像出力端末へ送信されるメタビットデータにより選択的に制御されるオブジェクト最適化画像形成変調サブシステムを含む画像処理システムを有し、
前記メタビットは、前記バイトの画像が対応するオブジェクトのタイプにより異なる駆動信号セットポイント間で選択するように前記オブジェクト最適化画像形成変調サブシステムを制御し、前記駆動信号セットポイントが前記バイトの画像データに対応する前記オブジェクトの前記形成に影響を与える、プリンタコントローラ。
【請求項３】　画像形成デバイスへ組み込まれ、前記画像形成デバイスが、ファクシミリマシーン、プリンタ、ディジタル複写機、及びラスター出力スキャナから成るグループから選択される、請求項２に記載のプリンタコントローラ。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、プリント装置において、最大又は飽和レーザ又はLEDパワーを調節するための方法及び装置に関する。より詳細には、本発明は、現在露光されている画像オブジェクトのタイプに基づいて、露光レベルを制御するための方法に関する。
【０００２】
【従来の技術】ディジタルカラーワークステーション、複写機及びプリンタの発展に伴って、ページ画像を合成しプリントするためにグラフィクアーチストに以前より依存していたページ画像のクリエータ（作成者）は、代わりに、ディジタルカラー複写機／プリンタに接続されたコンピュータを使用して、自分自身でページ画像を生成し、合成し、プリントする事ができる。
【０００３】しかしながら、ページ画像を生成するため、ページ画像をプリントエンジン命令に分解するため、及びプリントエンジンを制御するためのこのような従来技術のディジタルシステムは、ページ画像を単一で

2

一体の画像として処理していた。このように、テキスト／ラインアートのために最適化されるページ画像において、高周波ハーフトーンスクリーンが使用される場合、そのページ画像のテキスト部分は、きわめてシャープである。しかしながら、ページ画像の一定のカラー部分は、プリンタノイズからの明瞭な斑化を含む。更に、ページ画像のサンプリングされたカラー部分とスイープ（掃引）部分は、高周波スクリーンで利用され得る十分なグレイレベルの欠乏により明瞭な輪郭形成を含む。
【０００４】大きな一定のカラー部分のために最適化されたページ画像において、プリンタの不安定性を隠すために特に設計されたハーフトーンスクリーンは、高品質で、テキストが無く、アーチファクトの無い一定のカラー領域を生成する。しかしながら、テキストのシャープネスが低下され、各ティント毎のグレイ値は、あまり関連せず、サンプリングされたカラー部分とスイープ部分は、許容できない。そのグレイレベルは、各ドットレベルが他のレベルと関係なく別々に設計されるので、一つのレベルからその次のレベルへ滑らかにステップしない。
【０００５】サンプリングされたカラー部分とスイープ部分に対して最適化されたページ画像において、スイープ部分とサンプリングされたカラー部分は、低周波ハーフトーンスクリーンがより多く利用できるグレイレベルで使用されるので、より高い品質を示す。しかしながら、テキストは、低品質で表され、且つ一定のカラー部分は、明瞭な模様化を示す。
【０００６】しかしながら、各画像を単一のビットマップ画像又はバイトマップ画像として処理した従来のシステムにおいて、あらゆる一つのタイプのオブジェクトに対して画像を最適化することは、他のタイプのオブジェクトの画像品質と妥協することが必要であった。従って、手作業による合成グラフィクアート分野におけるのと同様に、ページ画像を生成し、分解し、個別のオブジェクトのプリント特性が最適化できるプリントエンジンへ出力すると同時に、マイクロコンピュータを使用してページ画像を生成する時に利用できる利点と効率を維持するディジタルカラー複写機／プリンタ及び方法が従来技術では必要である。
【０００７】このようなページ画像は、PostScript™のようなページ記述言語（PDL），Interpress™、Windows™と共に使用されるようなGrAphicAl DispAy InterfAces（GDI）、Hewlett－PAckArd Printer CommAnd LAnguAge （PCL－５）、等を使用して生成される。
【０００８】市販されているレーザプリンタ製品は、一般的ではない強力なレーザパワーを使用して”白黒”画像を露光する事が知られている。これは、白画像と黒画像の間でのコントラストの増強のためになされ、この過露光による囲み領域は見る者にとって審美的に好まれる。この増強レーザパワーは、画像範囲の全てに提供さ

れる。

【０００９】しかしながら、この増強されたコントラストは、それに伴い、細いラインを生成することを不能にする。更に、テキスト及び／又はラインアートの過露光は、審美的に好まれるが、ハーフトーン画像の過露光は、増強されたレーザパワーがハーフトーンセルを過露光するので、不利である。ハーフトーンセルを過露光することは、ダイナミックレンジ全体に亘る制御を減少する。これは、レーザパワーの増強がプリントエンジンの全ダイナミックレンジに亘って形状関数の遷移に影響するために生ずる。特に、レーザパワーが増加されると、形状関数が素早く飽和され過ぎ、ダイナミックレンジのハイ端の中遷移できない。

【００１０】

【発明が解決しようとする課題】本発明は、PDL等を使用して定義されたページ画像に各タイプのオブジェクトを形成するために使用されるレーザパワーが最適に制御又は選択されるようにプリントデータ及びプリンタ制御コマンドを使用するプリンタコントローラ装置及び方法を提供することを課題とする。プリントデータ及びプリンタ制御コマンドは、PDLを使用して定義されたページ画像から変換されてもよいし、従来技術において公知の他のメカニズムにより提供されてもよい。

【００１１】本発明は、更に、ページ画像を形成する種々のオブジェクトのオブジェクトタイプに基づいて、レーザパワー情報を含む”メタビット”情報、即ち、各バイトの画像データを最も良くレンダリングする方法についての情報を生成し、このメタビットデータを画像出力端末（IOT）へ送るオブジェクト最適化プリンタ制御装置及び方法を提供する。

【００１２】本発明は、更に、ページ画像の各独立のオブジェクトに対するオブジェクトタイプに基づいて、自動的にレーザパワーを決定するオブジェクト最適化プリンタ制御装置及び方法を提供する。

【００１３】

【課題を解決するための手段】本発明と共に使用される装置及び方法のステップの例として、PDLを使用して記述され一連のPDL.コマンドとして格納されるページ画像は、オブジェクト最適化エレクトロニックサブシステム（OOESS）へ入力される。PDL分解手段は、PDLで記述されたページ画像をページ画像を表すデータ構造へ分解する。このデータ構造において、独立の画像オブジェクトに関連する情報は、保持される。この情報は、オブジェクトのタイプ、及び最大レーザ変調セットポイント、カラー、最適カラー空間、層（レイヤー）情報等の他の個別の特性に関連する情報を含む。

【００１４】一旦、PDL分解手段がPDL.で記述されたページ画像を変換すると、複写機／プリンタ、より一般的には、画像出力端末（IOT）、コマンド命令発生手段がページ画像のスキャンライン毎に、データ構造を一連のコマンド命令、カラー指定及びメタビットレンダリング命令へ変換する。メタビットは、各オブジェクトのタイプを決定するために各オブジェクトを解析するメタビット発生手段により自動的に発生されてもよいし、ページ画像のPDL記述の生成の間にページ画像クリエータにより明示的に設定されてもよい。ページ画像の各スキャンライン毎に、コマンド命令、カラー指定及びメタビットが発生されると、それらは、IOTコントローラへ出力される。

【００１５】IOTコントローラは、分解処理の間に発生されたコマンド命令、カラー指定及びメタビットを受信する。IOTコントローラは、一定カラー及びサンプリングされたカラーデータを結合し、それをメタビットデータと共にIOTへ送信する。

【００１６】IOTの第１の例示の実施の形態において、IOTは、二つ以上のハーフトーンスクリーン発生器、閾値回路、カラー空間変換回路、及びレーザパワーセクション回路を含むことができる。IOTコントローラから出力されるバイト幅カラーデータ及びメタビットは、IOTへ入力される。メタビットは、カラーデータに対してどのハーフトーン発生器又は閾値回路が使用されるか、及びどのカラー変換がカラー空間変換回路によって適用されるかを決定するために使用される。また、メタビットは、画像データをプリントするために使用されるべき択一的なレーザパワーセットポイント、即ち、最大レーザ強度、を選択するために使用される。一旦、IOTがカラーデータ及びメタビットからラスターデータを発生すると、レーザ強度データを含むラスターデータは、ページ画像を出力シート上に形成するためにマーキングサブシステムへ出力する。

【００１７】或いは、サンプルカラーデータ圧縮／圧縮解除回路、マスクデータ圧縮／圧縮解除回路、一定カラー圧縮／圧縮解除回路、及びコマンドデータ圧縮／圧縮解除回路が、バスインターフェースと統合コンバナ（結合器）のIOTコントローラに組み込まれてもよい。従って、これらの圧縮／圧縮解除回路は、システムバス上でのIOTコントローラへの転送のために、データ構造の種々のタイプのデータが最適に圧縮されることができる。更に、IOTコントローラの第１の例示の実施の形態において、カラー空間変換器は、IOTコントローラへ組み込まれる。カラー空間変換器をIOTコントローラへ組み込むことによって、ページ画像の異なるオブジェクトに対するカラー空間変換は、丁度プリント前に実行されることができ、更に、オブジェクトタイプに基づいて、最適化され得る。更に、カラー空間変換器をIOTコントローラへ組み込むことは、本発明のシステムと共に使用され得るIOTの範囲を広げる。或いは、これらの圧縮／圧縮解除回路に対する他の形態が使用されてもよい。

【００１８】更に、二つのカラー空間変換器は、IOTへ出力されるカラーデータを発生するために使用され得る。このように、四つのカラー層分離、C、Y、M、Kの各々を表す、４バイトのデータが同時に出力されてもよい。全４カラー層分離に対するカラーデータを同時に出力することにより、IOTに対するより高いページ出力が達成され、且つIOTのより広い範囲が本発明のシステムと共に使用され得る。

【００１９】本発明の具体的態様として、第１の態様は、異なるタイプのデータを含む画像データのバイト間を区別するステップと、第１のステップポイントでレーザパワー駆動信号を使用して第１のタイプのバイト画像データをプリントするステップと、第２のステップポイントで前記レーザパワー駆動信号を使用して第２のタイプのバイトの画像データをプリントするステップと、を備え、前記第１のステップポイントが前記第２のステップポイントと異なっている、プリント方法である。

【００２０】第２の態様は、バイトの画像データに対応するオブジェクトのタイプに基づいてそのバイトの画像データをプリントするために使用される駆動信号セットポイントを調節するプリンタコントローラであって、プリントデータ及びメタビットデータを受信する画像出力端末を備え、前記画像出力端末が前記画像出力端末へ送信されるメタビットデータにより選択的に制御されるオブジェクト最適化画像形成変調サブシステムを含む画像処理システムを有し、前記メタビットは、前記バイトの画像が対応するオブジェクトのタイプにより異なる駆動信号セットポイント間で選択するように前記オブジェクト最適化画像形成変調サブシステムを制御し、前記駆動信号セットポイントが前記バイトの画像データに対応する前記オブジェクトの前記形成に影響を与える、プリンタコントローラである。

【００２１】第３の態様は、第２の態様において、プリンタコントローラが、画像形成デバイスへ組み込まれ、前記画像形成デバイスが、ファクシミリマシーン、プリンタ、ディジタル複写機、及びラスター出力スキャナから成るグループから選択される。

【００２２】

【発明の実施の形態】上述のように、ページ記述言語（PDL）で記述されたページ画像を分解するための従来の処理は、クリエータ（生成者）がページ画像全体に亘ってプリンタ依存ハーフトーンを明示的に挿入しない限り、PDLを使用して記述されたページ画像を分解することによって発生されるビットマップ又はバイトマップをプリントする時に妥協される必要がある。以下の記述は、PDLに対してのみ参照されるが、PDLと等価な上述のあらゆるシステムが以下に述べられるオブジェクト最適化レンダリングシステム及び方法と共に使用され得る。

【００２３】即ち、上述のように、PDLを使用して記述されたページ画像を分解及びプリントするための従来のシステムにおいて、ページ画像から成る種々のオブジェクトは、PDL座標からデバイス座標へ変換され、ストロークが厚く太くされ、オブジェクトが夫々スキャンライン上のオブジェクトの位置に対応する一連のボックスへ変換され、これらのボックスは、バイトマップ（又は画像が黒／白）の場合は、ビットマップ）へロードされる。ページ画像のオブジェクトを分解することによって形成されたボックスをバイトマップへ書き込むことによって、オブジェクトの異なるタイプ間の全ての区別が失われる。

【００２４】これとは反対に、本発明において、PDLにより記述されたページ画像は、ページ画像より成るオブジェクトタイプ間の区別が維持されるように分解される。ページ画像内の異なるオブジェクトのオブジェクトタイプを維持することによって、異なるオブジェクトタイプの処理は、オブジェクトのそのタイプへ最適化できる。

【００２５】従来より、連続的に変化する濃度の背景を形成するために粗いスクリーンハーフトーン発生処理が利用されると同時にシャープなエッジのテキスト画像を形成するために微細なハーフトーン発生処理が利用された。オブジェクトの各タイプ毎に、最適ハーフトーン発生処理を利用することによって、背景の連続的に変化する光学的濃度が維持され得る。同時に、テキスト画像のシャープなエッジを維持するための最適ハーフトーン発生処理が維持され得る。

【００２６】このように、高周波ハーフトーンスクリーンが最適化されたテキスト画像を形成するために使用され得ると共に、色合い（tint）指定ハーフトーンセットが一定カラーの最適化されたブロックを形成するために使用され得ることが出来、更に低周波ハーフトーンスクリーンが最適化されサンプリングされたカラー画像及び掃引（スイープ）を形成するために使用され得る。

【００２７】ページ画像から成る種々のオブジェクトのオブジェクトタイプを維持することによって、オブジェクトの他の特徴は、ページ記述言語により使用されるカラー空間からIOTで使用されるシアン（C），マゼンタ（M），イエロー（Y）及びブラック（B）カラー空間へのオブジェクトのカラー空間変換のように、十分に最適化され得る。実際に、カラー変換処理の各ステップは、アンダーカラー除去処理及びトーン（階調）再生曲線処理のように、最適化され得る。同様に、圧縮処理は、オブジェクトにより最適化されることが出来、それにより、消費される記憶リソース（資源）及び分解されたページ画像に対する遷移ロードを最小にする。

【００２８】更に、要求される最適化の度合いに依存して、オブジェクトタイプの区別の異なるレベルが提供され得る。即ち、幾つかのインスタンス（例）において、オブジェクトが一定のカラーで着色されることを指示し、ハーフトーンスクリーン及びカラー空間変換のような全ての処理関数をこのレベルで最適化することで十分であろう。他のインスタンスにおいて、多くの可能なカ

ラー空間変換の一つ又は多くのハーフトーンスクリーン周波数又は角度の一つを指定するために、追加の区別のレベルが要求されてもよく、それらの全てが一定のカラーオブジェクトに適する。

【００２９】更に、処理関数は、画像内の各オブジェクトを形成するために使用されるレーザ光露光の適切なレベルを提供するために最適化される。優れたハーフトーンドットを形成するために、形状関数は、ハイライトからミッドトーンを介してシャドー（陰影）までの光露光のダイナミックレンジ全体に亘り一貫しなければならない。テキスト及び／又はラインアートが過露光される場合と同様にハーフトーンドットの過露光に起因する問題は、ハーフトーン形成において、形状関数がダイナミックレンジのミッドトーン部分へ移動すると、そのレンジで未露光領域が露光されないままにならない。従って、得られるハーフトーンが完全なオーバーラップとなり、形状関数が全ダイナミックレンジのまだ途中にある時に、シャドー領域に入る。従って、全ダイナミックレンジのハイエンドは、形状関数がダイナミックレンジのローエンドにある所からダイナミックレンジのハイエンドにある所まですでに遷移しているので、画像のプリント品質へ影響しない。

【００３０】しかしながら、テキスト及びラインアートの外観は、テキスト及び／又はラインアート領域が過露光されると、より好ましい。理由は、テキスト及び／又はラインアート領域がより高いパワーで露光されると、コントラストが強調されるからである。過露光がより良く見えるが、あらゆる画像内に非常に細いラインのレンダリングにおける精度を失うことになる。しかしながら、失われた精度は、テキスト及び／又はラインアートの改良された外観に比較して大きくない。従って、テキスト及び／又はラインアートの処理の間のレーザパワーをハーフトーンドットの処理の間のレーザパワーより大きくする事は有利である。

【００３１】テキスト及びラインアート生成とハーフトーン生成との間のこれらの差は、サイズ及び制御の問題に基づく。テキスト又はラインアートオブジェクト内のオブジェクトの個別の特徴のサイズは、ハーフトーンオブジェクト内の個別の特徴よりも大きい。従って、テキスト又はラインアートを形成することは、ハーフトーン処理により生成される画像よりも粗い画像を生成する。テキスト又はラインアートが過露光されると、個別の特徴があまり近接されずに離間されるので、過露光は、ハーフトーンドットが過露光される時と同程度までは形状関数を低下しない。更に、テキスト及びラインアートを形成することは、ハーフトーン形成と同様にトーンの制御を必要としない。ハーフトーン処理において、例えば、形状関数は、円形状からダイアモンド形状へ遷移し、ハーフトーンドットの画像濃度が、プリントエンジンの全ダイナミックレンジに沿ってハイライト濃度からミッドトーン濃度を介してシャドウ濃度へ遷移するに従って、円形状へ戻る。フルカラー画像において、これらの形状関数によってカバーされる領域の正確な制御は、例えば、マゼンタ、イエロー及びシアンによりカバーされる領域の比率が見る者によって知覚されるカラーを制御するので、重要である。

【００３２】図１に示されるように、画像出力端末（IOT）１７０によって使用できるラスターデータへ画像データのPDLフォームへ変換するためのオブジェクト最適化電子サブシステム（OOESS）１００は、好ましくはパーソナルコンピュータのような汎用コンピュータ、カリフォルニア州、マウンテンのSun Miycrosystems 社による製造されるSunSpArcTMのようなエンジニアリングワークステーション、ミニコンピュータ等によって提供される。OOESS１００は、PostScriptTMやInterPressTM互換ドキュメントを発生するプログラムのようなページ画像のPDL表現を発生するためのプログラム、ページ画像のGDI表現を発生するプログラム、やページ画像のグラフィカルコマンドセット表現を発生するプログラムのような内部PDLファイルソース手段を含むことができる。グラフィカルコマンドセットの一つのタイプは、レーザプリンタ及び／又はインクジェットプリンタを作動するために使用されるHewlett－PAckArd PCL－５のコマンドセットである。従って、用語”PDL”は、ページ画像のビット／バイトマップを発生する代わりに、ページ画像を記述するあらゆるタイプの表現を含むよう解釈されるべきであることが理解されるべきである。

【００３３】或いは、生成され直接分解されるというよりはむしろ、ページ画像のPDL表現は、不揮発性メモリやローカルネットワークやモデムを介してOOESS１００へ接続される遠隔地の汎用コンピュータのような何らかの遠隔PDLファイルソース手段１１２から受信されることができる。従って、ページ画像を表すPDLファイルがあらゆる従来のソースから得ることができることが理解されるべきである。

【００３４】ページ画像を表すPDLファイルがOOESS１００に一旦入力されると、それはバス１１４を介してメモリ１５０へ転送される。次に、PDLファイルは、PDL分解手段１３０により分解される。PDL分解手段１３０は、メモリ１５０に格納されたPDLファイルを読み出し、それを分解して図２８乃至図３１に示されるデータ構造を形成する。図２９に示されるデータ構造は、オブジェクトリストとオブジェクトタイプを指示するレンダリングタグを含む。PDL分解手段の動作は、以下で詳細に述べられる。

【００３５】PDL分解手段１３０が、PDL表現から発生されるページ画像の種々のオブジェクトを格納するメモリ１５０にデータ構造を発生すると、メモリ１５０に格納されたデータ構造がコマンド命令発生手段１４０により読み出される。コマンド命令発生手段１４０は、メモリ

１５０に格納されたデータ構造をスキャンライン毎にカラーデータ、ビットマスクデータ及びメタビットデータに対応する一連のコマンド命令に変換する。これらのコマンド命令、カラーデータ、ビットマスクデータ及びメタビットデータが、図３１に示されるように、メモリ１５０に格納される。

【００３６】図２に示されるように、IOTコントローラ１６０がバスインターフェース１６１０を介してバス１１４へ接続されることが予知される。OOESS１００の好ましい実施の形態において、SunSPArcTMワークステーションが使用される。このように、バス１１４は、SBus（Sバス）であり、バスインターフェース１６１０は、SBus１１４と共に働くよう設計され得る。しかしながら、バスインターフェース１６１０は、従来のパーソナルコンピュータ、SunSpArcTMのようなエンジリアリングワークステーション、マイクロコンピュータ等の何れにおいても使用され得る特定のバス１１４と共に働くように設計され得る。

【００３７】図２に示されるように、バスインターフェース１６１０は、バス１１４に対して３２ビット接続を提供する。このように、バスインターフェース１６１０は、各クロックサイクル間に４バイトワードを入力できる。好ましい実施の形態において、バスインターフェース１６１０は、連続するクロックサイクルで１６個の４バイトワードを連続的に読み出すことによって、６４バイトバーストで読み出すことができる。更に、好ましい実施の形態におけるバスインターフェース１６１０は、バス１１４を介してメモリ１５０への直接メモリアドレス（DMA）を提供する。

【００３８】バスインターフェース１６１０がメモリ１５０から４バイト部分のデータを受信すると、データは、５セットの先入れ先出し（FIFO）データレジスタの一つのセットに分配される。これら５つのFIFOは、サンプルカラーチャネルFIFO１６２０、マスクデータチャネルFIFO１６２２、一定カラーチャネルFIFO１６２４、メタビットチャネルFIFO１６２６、及びコマンド命令チャネルFIFO１６２８を含む。

【００３９】図２に示されるように、サンプルカラーチャネルFIFO１６２０は、FIFOの二つのバンクより成り、FIFOの各バンクが二つの５１２アドレスＸ９ビット幅FIFOより成る。このように、FIFOの各バンクの各FIFOは、バスインターフェース１６１０により受信された４バイトワードの１バイトを受信する。更に、バスインターフェース１６１０は、４個の追加のビットのデータを発生し、この追加のビットのデータの一つは、９番目のビットとして４個のFIFOの各々に格納される。これらの余分のビットの各々は、それに関連するバイトのデータが有効データであるか無効データであるかをフラグする（フラグを立てる）ために使用される。DMAデータの取り出しがワード境界のみになされるので、サンプリングされた画像に対する実際のデータがワード内で始まるということが時々起こる。この場合、実際のデータの開始に先行するワード内のバイトが無効としてフラグが立てられ、これらのバイトは、プリントされるのではなくて破棄され得る。

【００４０】マスクデータチャネルFIFO１６２２は、単一の２５６アドレスＸ１６ビット幅FIFOデータレジスタより成る。一定カラーチャネルFIFO１６２４は、２個のFIFOの１個のバンクより成り、バンクの各FIFOは、1個の２５６アドレスX9ビット幅FIFOデータレジスタよりなる。マスクFIFO１６２２及びカラーFIFO１６２４の各々が書き込みサイクル当り2バイトを格納できるので、バスインターフェース１６１０へ入力されるデータの単一４バイト転送を利用してデータを連続的にマスクFIFO１６２２及び一定カラーFIFO１６２４の両方へ提供できる。唯一の１６ビット幅内部バスがバスインターフェース１６１０をマスクFIFO１６２２、一定カラーFIFO１６２４及びコマンドFIFO１６２８へ接続し、唯一の８ビットバスがバスインターフェース１６１０をメタビットFIFO１６２６へ接続するので、２クロックサイクルが１個の４バイト転写をマスクFIFO１６６２、カラーFIFO１６２４及びコマンドFIFO１６２８へ書き込むために使用され、４クロックサイクルが４バイト転写をメタビットFIFO１６２６へ書き込むことが必要である。

【００４１】メタビットFIFO１６２６は、単一の５１２アドレスＸ８ビット幅FIFOデータレジスタから成る。コマンド命令FIFO１６２８は、ペアの５１２アドレスＸ８ビット幅FIFOデータレジスタの単一のバンクから成る。

【００４２】FIFO１６２０乃至１６２８の各出力は、マルチチャネルコンバナ（結合器）１６３０に接続される。マスクFIFO１６２２からの出力はシリアル化される。このように、唯一の単一ビット接続がマスクFIFO１６２２からマルチチャネルコンバイナ１６３０へ提供される。同様に、サンプルカラーFIFO１６２０及び一定カラーFIFO１６２４は2から1へ（２－ｔｏ－１）で多重化される。このように、サンプルカラーFIFO１６２０に対して、唯一の１８ライン内部接続がマルチチャネルコンバイナへ提供される。対の９ビット幅FIFOの各バンクは、そのデータを交互に出力する。同様に、９ビット幅接続のみが一定カラーFIFO１６２４とマルチチャネルコンバイナ１６３０との間に提供される。一定カラーFIFO１６２４の各９ビット幅FIFOはそのデータを交互にマルチチャネルコンバイナへ出力する。

【００４３】FIFO１６２０乃至１６２４とは対照的に、メタビットFIFO１６２６とコマンド命令FIFO１６２８は、図２に示されるように、夫々マルチコンバイナ１６３０へのフル幅接続が提供される。

【００４４】図３に関連して以下により十分に説明されるマルチチャネルコンバイナ１６３０は、コマンド命令FIFO１６２８から受信された命令及びマスクFIFO１６２

11

２から受信されたビットマップデータに基づいて、サンプルカラーチャネル１６２０及び一定カラーチャネル１６２４からのデータをデータFIFO１６４２への単一ストリームの出力へ結合する。

【００４５】データFIFO１６４２は、４KアドレスＸ９ビット幅FIFOデータレジスタから成る。マルチチャネルコンバイナがデータを８ビット幅バイトでデータFIFO１６４２へ出力するので、データFIFO１６４２の第９番目のビットが、メタビットFIFO１６２６から第１番目（最初）のビットを格納するのに使用される。また、マルチチャネルコンバイナ１６３０は、出力メタビットFIFO１６４０へ接続される。出力メタビットFIFO１６４０は、４KアドレスＸ３ビット幅FIFOデータレジスタよりなる。マルチチャネルコンバイナ１６３０は、各８ビットメタビットバイトを少なくとも二つの４ビット幅メタビットニブルへ分割する。各４ビットニブルの第１番目のビットが、データFIFO１６４２へ出力されると、各４ビットニブルの第２番目から第４番目のビットが出力メタビットFIFO１６４０へ出力される。実際に、マルチチャネルコンバイナから出力されるメタビットの数は、区別されるべきオブジェクトのタイプの数及び実行されるべきオブジェクトの各タイプのレベルの数に依存して、１、２、４、又は８であってよい。

【００４６】以下に述べられるように、各コマンド命令は、単一のスキャンラインにのみ関連し、そのスキャンラインの可変量を定義する。従って、マルチチャネルコンバイナの出力をIOTのページ及びラインへ適切に同期することが出来ないので、並行入力ポート１６５０がページ同期（sync）信号、ライン同期信号及びIOTクロック信号を受信してマルチチャネルコンバイナ１６３８へ入力する。

【００４７】出力FIFO１６４０及び１６４２は、並行出力ポートドライバ１６５２及び１６５４へ夫々接続される。これらの並行ポート１６５２及び１６５４からの出力がIOT１７０の入力ポートへ接続される。

【００４８】データFIFO１６４２が、プリント用に処理されるべきデータとしてIOT１７０により受信されることが理解されるべきである。しかしながら、本発明は、更に、データFIFO１６４２からの各バイトのプリントデータが出力メタビットFIFO１６４０からの１、２、４、又は８メタビットにより同期して伴われるということを提供する。これらのメタビットは、プリントされる前に各データバイトへなされるべき処理を指定する。説明されたように、この最適処理は、各データバイトが抽出されたオブジェクトに依存して、各データバイトと共に変化する。図３４に示されるように、IOTコントローラ１６０の好適な実施の形態において、IOT１７０は、IOT１７０へ送信されるメタビットデータにより選択的に制御されるオブジェクト最適化されたサブシステムの大きさを含む画像処理システム１７１０を有する。このように、メタビットチャネルは、各バイトのプリントデータに入ってくるオブジェクトに依存して異なるように処理させる。

12

【００４９】例えば、メタビットは、多数のハーフトーン発生器１７１２間で選択できる。一つのハーフトーン発生器は、高空間周波数ハーフトーン発生器を使用すべきカラーのテキストのようなオブジェクトからくると、タグ付けされたデータバイトと共に使用され得る。他の物は、写真や絵画等から得られるデータバイトと共に使用される低空間周波数高カラー解像ハーフトーン発生器であってもよい。各オブジェクト毎のハーフトーンスクリーン角度の制御のようなハーフトーン発生に対するより微細な制御は、メタビット選択可能である。ハーフトーンの選択に加えて、種々のオブジェクトのそれらのエッジで互いに対話する事に起因する問題を処理する、オブジェクト最適化カラー変換サブシステム１７１４、オブジェクト最適化トーン再生曲線変更サブシステム１７１６、オブジェクト最適化空間フィルタサブシステム１７１８及びオブジェクト最適化トラッピングサブシステム１７２０、及び異なるオブジェクトにおいて異なるように表れる種々のプリントシステムの問題を制御しようとするオブジェクト最適化エンジンノイズ抑制サブシステム、並びに他のオブジェクト最適化画像処理サブシステムのようなオブジェクトに依存する可変結果を生成するために、画像処理システム１７１０においてメタビットによって多くの追加の画像処理サブシステムが制御され得る。

【００５０】IOT１７０は、画像処理システム１７１０内にレーザパワー選択サブシステム１７２４を含む。メタビットは、画像処理システム１７１０のレーザ選択サブシステム１７２４を制御して、画像データの各特定のバイトに関連するメタビットに依存して、異なるセットポイントの間で選択する。特に、オブジェクト最適化レーザパワー選択サブシステム１７２４は、画像データのバイトがハーフトーンオブジェクト又はテキスト又はラインアートオブジェクト内に在るかに基づいて、そのバイトの画像データをレンダリングするために使用されるレーザパワーのセットポイントを変更する。オブジェクト最適化レーザパワー選択サブシステム１７２４は、オブジェクトをレンダリングするために最も適切なレーザパワーを提供するように複数のセットポイント間での選択を可能とする。

【００５１】例えば、メタビット値は、そのバイトの画像データがテキスト又はラインアートオブジェクトの部分であることを指示する場合、メタビット値がオブジェクト最適化レーザパワー選択サブシステム１７２４を制御してそのバイトの画像データを形成するために高いレーザパワーセットポイントを使用する。或いは、メタビット値は、そのバイトの画像データがハーフトーンオブジェクト内にあることを指示する場合、メタビット値がオブジェクト最適化レーザパワー選択サブシステム１７２４を制御してそのバイトの画像データを形成するために低いパワーセットポイントを使用する。

13

【００５２】これらのセットポイントの各々に関連するレーザパワーが他のセットポイントの関連において設定されても良いことが理解されるべきである。例えば、低いセットポイントは、あらゆる他のセットポイントから独立して設定される。次に、高いセットポイントは、低いセットポイントよりも１０％高く設定される。或いは、セットポイントが互いから独立して設定されてもよい。

【００５３】また、多くのプリンタが”書き込み黒”タイプであり、それらはテキスト／ラインアートのような現像されたマークであってもよいあるメディアの領域を露光することを理解すべきである。光がより高くなると、露光によりマークがより太くなる。他のプリンタは、マークを囲む領域を露光する”白書き込み”タイプであり、これらの露光領域が現像されず、未露光領域が現像される。白書き込みシステムにおいてより太いマークを作成することは、テキスト／ラインアートを囲む白領域により低い露光が必要である。

【００５４】更に、プリントシステムが入力スキャナのような他の同期データソースを有する場合、メタビットチャネルは、プリントに先立って、これらのデータストリームをデータマルチプレクサ１７３０を介してIOTコントローラ１６０から入ってくるプリントデータとミックスするために使用され得る。匹敵する装置がIOTコントローラ１６０においてミキシングを実行するために使用され得る。

【００５５】上述の本発明による装置の例示の実施の形態において、メタビットは、出力が予めロードされている１セットのメタビットマッピングレジスタ１７４０間で選択することにより、オブジェクト毎ベースで前記複数の画像処理サブシステム１７１２－１７３８を制御して、オブジェクト最適化画像処理モジュール及びプリントマルチプレクサ１７１２－１７３０の各々で指定されたサブセットの使用可能ラインを選択する。例えば、４メタビットが実行される場合、それらは、１セットの１６個のレジスタから選択して使用され得る。一方、これらのレジスタは、各々十分に制御でき、画像処理サブシステム及びプリントマルチプレクサ１７１２－１７３０のサブセットから選択されることが必要なあらゆるサイズのものであり得る。このように、各メタビット値の意味は、完全にプログラム可能であり、それが選択するレジスタの内容を変化することにより簡単に変更され得る。一方、そのレジスタは、そのメタビットによりタグ付けされるオブジェクトの特定のタイプに対して最も可能性の高いレンダリングを実行するために、画像処理サブシステム及びプリントマルチプレクサ１７１２－１７３０の完全なセットから選択してもよい。

14

【００５６】種々の画像処理サブシステム及びプリントマルチプレクサ１７１２－１７３０が交互にオブジェクト最適化プリントシステム全体に亘って、多くの他のポイントで表れてもよいことが理解されるべきである。例えば、図示されるように、オブジェクト最適化画像処理サブシステム及びプリントマルチプレクサ１７１２－１７３０をオブジェクトタイプが決定された後は何時でも、PDL分解手段１３０内、IOTコマンド命令発生手段１４０内のようなオブジェクト最適化ESS１００に、IOTコントローラ１６０の一部として、或いはIOT１７０内に配置することができる。また、これらのサブシステムをシステム全体に分散配置できる。

【００５７】更に、これらのオブジェクト最適化画像処理サブシステム及びプリントマルチプレクサ１７１２－１７３０は、ハードウェア又はソフトウェア或いはこれら二つの組み合わせで実行され得る。あらゆる場合において、サブシステムは、異なる手順及びそれらが処理しているオブジェクトのタイプに基づくデータを選択している。従って、異なるオブジェクトタイプに対して異なる処理結果を生成する。

【００５８】また、カラー測定デバイス１９０は、測定較正プリントと連結されるメタビット情報を使用して、オブジェクト最適化ベースで画像処理サブシステム１７１２－１７２２を自動的に調節し安定化することができる。例えば、図３５に示されるように、較正プリントを測定する出力カラー測定デバイス１９０は、較正プリントの実際の状態を指示する信号を出力する。メタビット情報及び出力カラー測定デバイスからの出力信号に基づいて、オブジェクト最適化出力訂正サブシステム１７３２は、オブジェクト最適化トーン再生曲線サブシステム１７１６における全体のセットから一つのトーン再生曲線（TRC）ルックアップテーブルを調節する。画像処理システム関数（機能）に対して行われる訂正は、オブジェクト最適化出力訂正サブシステム１７３２がメタビットチャネルでエンコードされたタグにより制御されるので、異なるオブジェクトに対して異なる。

【００５９】図３は、マルチチャネル１６３０の一つの例示の実施の形態を示す。図３に示されるように、メタビットFIFO１６２６は、メタビットアンパッカー１６３１に接続され、８ビット幅メタビットを少なくとも二つの４ビット幅メタビットニブルに変換する。メタビットアンパッカー１６３１からの出力は、四つのカラー／メタビットレジスタ１６３４－１６３７の各々に接続される。メタビットアンパッカー１６３１からの出力がFIFO制御及びコマンド／マスクチャネルプロセッサ１６８０により出力されるロードイネーブル（LE）信号に依り、カラー／メタビットレジスタ１６３４乃至１６３７の一つにロードされる。同様に、一定のカラーチャネルFIFO１６２４からのデータ入力は、４個のカラー／メタビットレジスタ１６３４乃至１６３７へ提供され、ロードイ

15 16

ネーブル信号に基づいてカラー／メタビットレジスタの一つにロードされる。

【００６０】サンプルチャネルFIFO１６２０からの１８ビット幅入力データは、サンプルチャネルアンパッカー１６３２Aに入力され、２バイト幅サンプルカラーデータ及び２ビット”有効ーバイト”データを２個の単一バイト出力に変換され、それらは、連続的に内部サンプルFIFO１６３２Bへ出力されるか、或いはそれらの対応する”有効バイト”はこのバイトが無効であるということを指示する場合は、破棄される。

【００６１】コマンド命令FIFO１６２８からの２バイト幅出力及びマスクFIFO１６２２からの１ビット幅出力は、コマンド／マスクチャネルプロセッサ１６８０へ直接入力される。また、コマンド／マスクチャネルプロセッサ１６８０へは、並行入力ポート１６５０を介してIOTから受信されるページ同期、ライン同期及びIOTクロックが入力される。コマンド／マスクチャネルプロセッサ１６８０制御信号をカラー／メタビットマルチプレクサ１６３３及び出力マルチプレクサ１６３９、及びFIFO１６２０－１６２８への４つの読出し信号を出力する。４つの読出し信号は、カラー／メタビット読出し（RD）信号、サンプル読出し信号、マスク読出し信号及びコマンド命令読出し信号よりなる。対応するFIFO１６２０－１６２８がコマンド／マスクチャネルプロセッサ１６８０からこれらの信号の一つを受信するとすぐ、対応するFIFO（単数又は複数）は、対応するチャネル上の次のデータを読み出す。FIFO（単数又は複数）１６２０乃至１６２８は、バスインターフェース１６１０によるドライ（dry）ランニングを防止し、計算された優先順位に従ってメモリ１５０に格納されたチャネルデータ構造（図３５を参照）から読み出されたデータでそれらを充填する。

【００６２】コマンドチャネルFIFO１６２８から受信されたコマンド命令に基づいて、コマンド／マスクチャネルプロセッサ１６８０は、カラー／メタビットレジスタ１６３４乃至１６３７の対応する一つに対してロードイネーブル信号ＬＥを発生する。コマンド／マスクチャネルプロセッサ１６８０の一つがイネーブルされると、同時にそれは、一定カラーFIFO１６２４からの９ビット幅一定カラーデータ及びメタビットアンパッカー１６３１＊からのメタビットデータをロードする。或いは、カラーデータは、別個のコマンドでメタビットから独立してロードされてもよい。しかしながら、カラー／メタビットレジスタ１６３４乃至１６３７は、それらが絶えず同時にロードされるので、何時も共に参照される個別のレジスタであることが好ましい。

【００６３】また、コマンド／マスクチャネルプロセッサ１６８０は、出力マルチプレクサ１６３９及びメタビットレジスタ１６３８へ入力されるべきカラー／メタビットレジスタ１６３４乃至１６３７の一つを選択するために、カラー／メタビットマルチプレクサ１６３３に対して制御信号を発生する。更に、コマンド／マスクチャネルプロセッサ１６８０は、カラー／メタビットマルチプレクサ１６３３からの出力と内部サンプルカラーFIFO１６３２Bからの出力との間で選択するために出力マルチプレクサ１６３９に対して制御信号を発生する。出力マルチプレクサ１６３９は、カラー／メタビットマルチプレクサ１６３３により出力される８ビット幅カラーデータ及び第１番目のビットのメタデータを出力する。これは、データFIFO１６４２へ入力される。同時に、メタビットレジスタ１６３８は、第２番目－第４番目のメタビットを格納し、それらを出力メタビットFIFO１６４０へ出力する。好ましい実施の形態におけるように、４ビット幅メタビットニブルの全てを含むことは必要ない事が理解されるべきである。或いは、IOTへ提供されるメタビットの数は、１、２、又は４、或いは８であってもよい。このように、図２及び３に示される例示の実施の形態は、IOTへ提供されるメタビットの数を４に制限するものとして解釈されるべきではない。

【００６４】図４は、コマンド／マスクチャネルプロセッサ１６８０の内部機能ブロックを示す。図４において、コマンドチャネルライン及びマスクデータチャネルラインを除いて、制御ラインのみが示されている。コマンド／マスクチャネルプロセッサ１６８０において、二つの構成レジスタ１６８１及び１６８４、レジスタCSR０及びCSR１，が夫々設けられている。第１の構成レジスタ１６８１は、表１に示される特徴（機能）の制御を提供する。

【表１】

ビット	ハイ	ロー
8	将来の使用のためにリザーブ	将来の使用のためにリザーブ
7	白=FF	白=00
6	ノーマルモード	診断モード
5	レジスタ読出し	FIFO読出し
4	FIFOﾓｰﾄﾞに対してﾚｼﾞｽﾀをオン	FIFOﾓｰﾄﾞに対してﾚｼﾞｽﾀをオフ
3	メタビットFIFOを使用	ｶﾗｰFIFOの9番目のﾋﾞｯﾄを使用
2	メタビットアンパック1をオン	メタビットアンパック1をオフ
1	メタビットアンパック0をオン	メタビットアンパック1をオフ

構成／ステータスレジスタ０(CSR０)ビット割り当て

【００６５】第１の構成レジスタ１６８１のビット１及びビット２は、メタビットをアンパックするためにメタビットアンパッカーにより使用される２つ（必要に応じて又は３或いは４）メタビットアンパッキング方式を定義する。即ち、メタビットデータのバイトがどのようにメタビットパケットへ分解されるかを決定するための追加の方法が必要な場合、第１および第２のビットが共にメタビットアンパッキングの４つまでのタイプを提供するために使用され得る。

17 18

【００６６】第１の構成レジスタ１６８１のビット３は、唯一つの１ビット幅メタビットチャネルがIOTへ提供される場合に、メタビットFIFOをディスイネーブル（使用不能）にするために使用される。第１の構成レジスタ１６８１のビット４及びビット５は、診断モードで使用されてデータをFIFO及び内部レジスタから読み出す。第１の構成レジスタ１６８１のビット６は、マルチチャネルコンバイナがノーマルモードか診断モードかを指示するために使用される。第１の構成レジスタ１６８１のビット７は、白（例えば、紙の自然なバックグラウンドカラー）が提供されるのが全て０から成るデータバイトか全て１から成るデータバイトかを指示するために使用される。第１の構成レジスタ１６８１の第８番目のビットは実行されない。

【００６７】第１の構成レジスタ１６８１のビット６がローの時に、マルチチャネルコンバイナ１６３０は、診断モードである。診断モードにおいて、マルチチャネルコンバイナ１６３０の内部レジスタの何れも審査され得る。例えば、カラー／メタビットレジスタ１６３４乃至１６３７及びサンプルFIFO１６３２Bの内容をリアルタイムで審査できる。表２に示される診断SELラインを０－４の値へ設定することによって、SELラインは、選択されたレジスタの値を強制的にデータFIFO１６４２へ送ることができる。表２に示されるようにSELラインの値５－１４は、更なる診断情報のためにマルチチャネルコンバイナ１６３０の他のレジスタの読出し及び書込みのために使用される。

【表２】

SEL	アクセスされるレジスタ
0	ﾋﾞﾃﾞｵFIFOへのｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ0
1	ﾋﾞﾃﾞｵFIFOへのｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ1
2	ﾋﾞﾃﾞｵFIFOへのｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ2
3	ﾋﾞﾃﾞｵFIFOへのｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ3
4	ﾋﾞﾃﾞｵFIFOへのｻﾝﾌﾟﾙﾏｽｸﾚｼﾞｽﾀ
5	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配されるSR0
6	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配されるSR1
7	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配される又は書込まれるCSR0
8	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配される又は書込まれるCSR1
9	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配されるﾒﾀﾋﾞｯﾄ出力ﾚｼﾞｽﾀ
10	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配される又は書込まれるﾋﾞﾃﾞｵﾃﾞｰﾀｶｳﾝﾀﾛｰ（0:7）
11	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配される又は書込まれるﾋﾞﾃﾞｵﾃﾞｰﾀｶｳﾝﾀﾊｲ（8:12）
12	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配される又は書込まれる先端白ｶｳﾝﾀ（0:7）
13	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配される又は書込まれる後端白ｶｳﾝﾀ（0:7）
14	ｻｰﾄﾞﾊﾟｯﾀｰﾝﾊﾞｽに配されるﾌｫﾝﾄﾀﾞｳﾝﾛｰﾄﾞﾚｼﾞｽﾀ
15	将来使用のためにﾘｻﾞｰﾌﾞ

診断SEL値に対するレジスタアクセスデコード表

【００６８】構成レジスタ１６８１のビット６がハイ（即ち、ノーマルモード）の時、マルチチャネルコンバイナ１６３０は、ノーマルプリントデータを発生し、それをデータFIFO１６４２へ送信される。

【００６９】表３は、第２の構成レジスタ１６８４のビット割り当てを示す。ビット７及びビット８は、IOTイ※ンターフェース信号の極性を決定することによってIOTインターフェースにフレキシビリティを提供する。ビット５及びビット６は診断支援を提供する。ビット１、２、３及び４は、この例示の実施の形態では実行されない。

【表３】

ビット	ハイ	ロー
8	Ｐ同期立下り	Ｐ同期立ち上がり
7	Ｌ同期立下り	Ｌ同期立ち上がり
6	偽Ｐ同期オン	偽Ｐ同期オフ
5	偽Ｌ同期オン	偽Ｌ同期オフ
4	将来使用のためリザーブ	将来使用のためリザーブ
3	将来使用のためリザーブ	将来使用のためリザーブ
2	将来使用のためリザーブ	将来使用のためリザーブ
1	将来使用のためリザーブ	将来使用のためリザーブ

構成／ステータスレジスタ１(CSR1)ビット割り当て

【００７０】再び図４を参照すると、コマンド／マスクコントローラ１６８６は、１６ビットコマンド命令及び１ビットマスクチャネル入力を受信する。表４及び５は、ノーマル／リピートモードビットがノーマルにセットされる（即ち、ビット１５が０へセットされる）と、１６ビットコマンド命令に対するビット割り当てを示す。

【表４】

19 20

【表５】

コマンドビット	フィールド記述子
0	RepCnt0
1	RepCnt1
2	RepCnt2
3	RepCnt3
4	RepCnt4
5	RepCnt5
6	SCRes0
7	SCRes1
8	SrcB0
9	SrcB1
10	SrcA0
11	SrcA1
12	Cntr10
13	Cntr11
14	LdColor
15	Repeat/Normal

ノーマルモード（ビット15=0）

20

	LdColor	コマンド
	0	カラーをロードしない
	1	SrcB CC/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀをFIFOｷﾞからﾛｰﾄﾞ 注:1）"ﾒﾀﾋﾞｯﾄﾓｰﾄﾞ"を使用の場合、 ﾒﾀﾋﾞｯﾄがｱﾝﾊﾟｯｶｰからﾛｰﾄﾞされる。 注:2）ｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀは、これがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ0乃至2に対するｺﾏﾝﾄﾞの第1のｸﾛｯｸｻｲｸﾙ又はｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ3のこのｺﾏﾝﾄﾞのあらゆるｸﾛｯｸｻｲｸﾙである時、ﾛｰﾄﾞされる。
Cntrl1	Cntr10	コマンド
0	0	常にSrcB
0	1	常にSrcA
1	0	ﾏｽｸがSrcAとSrcBとの間で選択
1	1	ﾏｽｸがSrcBとSrcAとの間で選択
SrcA1	SrcA0	コマンド
0	0	SrcAがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ0を選択
0	1	SrcAがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ1を選択
1	0	SrcAがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ2を選択
1	1	SrcAがｻﾝﾌﾟﾙを選択
SrcB1	SrcB0	コマンド
0	0	SrcBがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ0を選択
0	1	SrcBがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ1を選択
1	0	SrcBがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ2を選択
1	1	SrcBがｶﾗｰ/ﾒﾀﾋﾞｯﾄﾚｼﾞｽﾀ3を選択
SCRes1	SCRes0	コマンド
0	0	ｻﾝﾌﾟﾙｸﾛｯｸを4で割る
0	1	ｻﾝﾌﾟﾙｸﾛｯｸを3で割る
1	0	ｻﾝﾌﾟﾙｸﾛｯｸを2で割る
1	1	ｻﾝﾌﾟﾙｸﾛｯｸを1で割る
	RepCnt0-5	コマンド
		ﾉｰﾏﾙｺﾏﾝﾄﾞﾘﾋﾟｰﾄｶｳﾝﾄ

ノーマルコマンドデコード表

【００７１】ノーマルモードにおいて、表４に示される　50　ように、ビット０－５は、６ビットカウント値を表す。

21 22

カウント値は、このコマンドが連続するピクセルの幾つ有効かを指示する。表５は、ビット６及び７に対する値に依存して、サンプルクロックが１、２、３又は４で割られる。サンプルクロックを割ることによって、出力画像の解像度が内部サンプルFIFO１６３２Bの現在のカラーを繰り返し出力させることにより変化され得る。即ち、"サンプルクロックを４で割れ"コマンドが選択されるようなビット６及び７の値が共に０の時、内部サンプルFIFO１６３２Bが各クロックサイクル毎に４回繰り返し出力される。従って、例えば、４００スポット／インチ（spi）解像度のプリンタは、何ら追加の処理が必要無く、サンプルマップを４００spi、２００spi、１３３spi、及び１００spiでプリントすることができる。

【００７２】ビット８及び９は、カラーレジスタ選択B値を表し、またビット１０及び１１は、カラーレジスタ選択A値を表す。表５に示されるように、カラーレジスタ選択A値及びカラーレジスタ選択B値は、カラー／メタビットレジスタ１６３４－１６３７（又は内部サンプルFIFO１６３２B）の何れがデータFIFO１６４２へコンバイナによってパスされるかを指示する。カラーレジスタ選択A値がカラー／メガビットレジスタ１６３４、１６３５及び１６３６又は内部サンプルFIFOの間で選択し、且つカラーレジスタ選択B値がカラー／メタビットレジスタ１６３４ー１６３７間で選択する．二つの選択値A及びBがあり、それにより１ビットチャネルがそれらの間の切り替えのために使用される。このモードは、ビット１２及び１３により指定され、且つなんらの追加の命令オーバーヘッド無しにA及びBにより指定される二つのレジスタの間で迅速に切り換えるために使用される。

【００７３】ビット１２及び１３は２ビットコンバイナ制御値を表す。コンバイナ制御値は、６ビットカウント値によって指示される次のn個のピクセルに対してマルチチャネルコンバイナ１６３０によってデータFIFO１６４２へ出力されるべきカラーデータを指示する。表５に示されるように、コンバイナ制御値は、出力されるべきカラーがカラーレジスタ選択A値、又は、カラーレジスタ選択B値によって選択されるレジスタ、又はマスクチャネルによって更に制御されるその二つのレジスタの組み合わせから来るのかを指示する。

【００７４】最後に、表５に示されるように、ビット１４は、カラーチャネルFIFO１６２４から出力される次のカラーデータがカラーレジスタ選択B値によってカラー／メタビットレジスタ１６３４－１６３７の一つにロードされるべきであるか否かを指示する。

【００７５】動作において、新たなノーマルモードコマンドがコマンドチャネルFIFO１６２８から入力されると、ロードカラービット１４は、カラーチャネルFIFO１６２４に格納された次の８ビットカラーデータバイトがカラーレジスタ選択Bビット８及び９によって指示されるカラー／メタビットレジスタ１６３４－１６３７にロードされるべきか否かを決定するためにデコードされる。カラーレジスタ選択Bビット８及び９がカラー／メタビットレジスタ１６３４－１６３６（カラー／メタビットレジスタ０－２）を指示する時、カラーは、コマンドの第１のクロックサイクルでのみカラーチャネルFIFO１６２４からロードされる。

【００７６】しかしながら、カラーレジスタ選択Bビット８及び９がカラー／メタビットレジスタ１６３７（カラー／メタビットレジスタ３）を指示すると、新たな一定カラーデータがこのコマンドの各クロックサイクル毎に、カラーチャネルFIFO１６２４からカラー／メタビットレジスタ１６３７にロードされる。このように、実際に、一定カラーチャネルが低速サンプルチャネルとして使用され得る。幾つかの複雑性の高い画像において、スキャンラインにおける一定カラーデータは、迅速に変化するので（即ち、１－８画素の全てを変化する）ので、新たなカラー毎に新たなコマンドワードを発生し、格納し、伝送し、そしてデコードするために必要な処理が過剰になり、その結果、バイトマップ表現に相対して負の圧縮となる。

【００７７】この効果的な負の圧縮は、一定カラーのストリングを一定カラーチャネルFIFO１６２４へ提供し、カラーFIFO１６２４から一定カラーのストリングの一つの一定カラーバイトをクロックサイクル毎に読出し、それをカラー／メタビットレジスタ１６３７へロードすることにより回避され得る。サンプリングされたデータチャネルFIFO１６２０におけるように、一定カラーチャネルFIFO１６２４が一回に４バイトではなくて、１回に２バイトをロードできるに過ぎないので、このように一定カラーチャネルFIFO１６２４を使用することは、一定カラーチャネルのスループットがサンプルデータチャネルのスループットより低くなる。しかしながら、この"スローサンプル"方法は、レギュラーサンプルデータチャネル（以下に詳述される）に使用されるDMAポインタを発生する時に必要なオーバーヘッドをさける。このように、この"スローサンプル"チャネルは、迅速に変化する一定カラーのショートストレッチに対しては最も有用であるが、大きな画像に対しては使用されない。

【００７８】次に、コンバイナ制御ビット１２及び１３は、カラーデータソースとしてカラー／メタビットレジスタ１６３４－１６３７（又は内部サンプルFIFO１６３２B）の何れが使用されるかを決定するためにカラー選択レジスタバンクA及びBの何れが使用されるかを決定すると共に、マスクチャネルからのマスクデータがマルチチャネルコンバイナ１６３０からデータFIFO１６４２へのデータフローを更に制御するために使用されるか否かを決定するためにデコードされる。勿論、ロードカラービット１４及びカラー選択レジスタBビット８と９が"スローサンプル"カラーチャネルとしてカラー／メタビットレジスタ１６３７を使用することを指示すると、制

23

御ビット１２と１３は、カラーレジスタ選択Bバンクを使用することを指示しなければならない。他方、コマンドビット１２と１３がカラーデータソースを指示するためにカラーレジスタ選択バンクAを使用することを指示すると、ロードカラービット１４及びカラーレジスタ選択バンクBビット８と９は、カラーレジスタ選択Aビット１０と１１によって指示されるものではなくてカラー／メタビットレジスタ１６３４－１６３７の一つへ一定カラーチャネルFIFO１６２４から出力される次のカラーデータワードをプリロードするために使用され得る。このように、次の一定カラーデータワードは、カラー／メタビットレジスタの一つへプリロードされ得る。

【００７９】次に、レジスタ選択バンクAとBの内の何れがコンバイナ制御ビット１３と１４によって指示されるかに依存して、カラーレジスタ選択Aビット１０と１１又はカラーレジスタ選択Bビット８と８が出力されるカラーデータソースとしてカラー／メタビットレジスタ１６３４－１６３７（又は内部サンプルFIFO１６３２B）の何れが選択されるかを決定するためにデコードされる。次に、コンバイナ制御ビット１２と１３がカラーレジスタ選択バンクAを指示し、カラーレジスタ選択Aビット１０と１１が内部サンプルFIFO１６３２Bに格納されたデータを使用することを指示すると、サンプリングされたカラー解像度ビット６と７は、サンプルクロックを

24

分割するファクタを決定するためにデコードされる。次に、コンバイナ制御ビット１２と１３が１ビットマスクチャネルを使用して出力カラーを更に制御することを指示すると、データFIFO１６４２への出力は、選択Aビット１０と１１によって選択されるカラーレジスタ（又はサンプルFIFO）と、選択Bビット８と９によって選択されるカラーレジスタとの間で切り換えられる。最後に、繰り返しカウントビット０－５が現在のコマンド命令は、幾つ画素が有効であるかを決定するためにデコードされる。

【００８０】従って、図４に示されるように、カラー／メタビットレジスタロード選択１６８５は、ロードカラービット１４及びカラーレジスタ選択Bビット８と９の値によって制御される。同様に、カラー／メタビットマルチプレクサ選択コントローラ１６８９と出力マルチプレクサ選択コントローラ１６９２は、上述のように、コマンド制御ビット１２と１３、カラーレジスタ選択Bビット８と９、カラーレジスタ選択Aビット１０と１１、及びマスクチャネルの組み合わせによって制御される。

【００８１】しかしながら、繰り返しモードビット１５がハイにセットされると、コマンド命令のコマンドビットが表６と７に指示される形態を取る。

【表６】

コマンドビット	フィールド記述子
0	RepCnt0
1	RepCnt1
2	RepCnt2
3	RepCnt3
4	RepCnt4
5	RepCnt5
6	RepCnt6
7	RepCnt7
8	RepCnt8
9	RepCnt9
10	RepCnt10
11	RepCnt11
12	RepCnt12/Mask Scanline Disable
13	RepType0
14	RepType1
15	Repeat/Normal

繰り返しモード（ビット15-1）

【表７】

25 26

RepType1	RepType0	コマンド
0	0	最後のﾉｰﾏﾙｺﾏﾝﾄﾞを繰り返せ
0	1	白出力：先端白ﾃﾞｰﾀのために使用。繰り返しｶｳﾝﾄが0に等しくあるべきという点に注意。
1	0	ﾗｲﾝ端(End of Line)：後端白ﾃﾞｰﾀのために使用，繰り返しｶｳﾝﾄが0に等しくあるべきという点に注意。
1	1	ﾍﾟｰｼﾞ端(End of Page)
	RepI2	コマンド
	0	注：ﾗｲﾝ端がこのｺﾏﾝﾄﾞからﾃﾞｺｰﾄﾞされると、Rep12は，Mask Channel Scanline Disable; もしこのﾋﾞｯﾄが0の時，Mask Channelは次のｽｷｬﾝﾗｲﾝに対して使用不能である。
	1	ﾗｲﾝ端の時，Mask Channelは，次のｽｷｬﾝﾗｲﾝに対して使用不能である。
	RepCnt0-12	コマンド
		ｺﾏﾝﾄﾞ繰り返しｶｳﾝﾄ(Command Repeat Count)を繰り返せ。

繰り返しコマンドデコード表

【００８２】表６及び７に示されるように、コマンドビット０－１２は、繰り返しカウントを表す。繰り返しモードにおける繰り返しカウントは、ノーマルモードにおいて繰り返しカウントの範囲を１２８回提供するので、繰り返しモードは、スキャンラインの非常に幅広い構造がスキャンされた画像又は大きな一定カラー領域のような同じカラーデータ又はオブジェクトを持つ時に使用される。

【００８３】最後に、ビット１３と１４は、繰り返しモードコマンド命令の繰り返しタイプを表す。繰り返しタイプがノーマルである場合、前のコマンドは、繰り返しカウントビット０－１１によって指示される追加のピクセル数に対して繰り返される。この場合、図４に示される前のコマンドレジスタ１６８２が使用される。繰り返しコマンドタイプが先端白データ又はライン後白データの端を指示すると、一つの単一白バイトが白（透明）発生器１６９０によってデータFIFO１６４２へロードされる。この白（透明）発生器１６９０は、構成レジスタ１６８１によって白の訂正値へセットされる。繰り返しコマンドタイプがページの端を指示すると、このコマンドは、回路がリセットされるまで、更なるコマンドの実行を停止する。

【００８４】繰り返しタイプビット１３と１４がラインタイプの端を指示した時にのみ、ビット１２は、マスクスキャンラインイネーブル／ディスイネーブルビットであるように再定義され、そのビットは、次のスキャンラインに対してマスクチャネルをオン又はオフするために使用されることが理解されるべきである。これにより、これらの関連するマスクデータを有するスキャンラインのみがマスクチャネルに配され、それはマスクチャネルに必要なデータを圧縮する。これは、表７に示されている。

【００８５】繰り返しタイプビット１３と１４が出力白タイプ又はラインタイプの端を指示すると、繰り返しカウントは使用されないことも理解されるべきである。代わりに、ラインの初め及び終わりの白スペースは、実際のページデータを発生するために利用できる時間量を増加する利点のあるビデオデータ発生の特別の場合である。先端及び後端白（透明）データを発生するために、三つのカウンタがIOTインターフェースコントローラ１６９１に取り付けられ、これらは、各カラー分離の初めに正確なデータがプリロードされる。先端白カウンタ１６９４は、各スキャンラインの開始マージンをカウントするために使用される。ライン及びページ同期（sync）ハンドラー１６８８が新たなライン同期信号をIOTから受信するとカウントを開始する。先端白カウンタ１６９４がカウントしている間に、IOTインターフェースコントローラ１６９１は、データFIFO１６４２の読出しイネーブルを使用不能にして先端白繰り返しコマンドによってデータFIFO１６４２に配された単一の白バイトを繰り返し出力する。この白バイトは、IOT１７０によって繰り返し読み出される。先端白カウンタ１６９４が0に達すると、ビデオデータカウンタ１６９５は、カウントを開始し、IOTインターフェースコントローラ１６９１は、データFIFO１６４２及び出力メタビットFIFO１６４０をイネーブル（使用可能）にしてIOTコントローラ１６９１からの同期下でそれらのデータをIOT１７０へ流す。ビデオデータカウンタ１６９５が0に達すると、後端白カウンタ１６９６は、カウントを開始し、IOTインターフェースコントローラ１６９１は、データFIFO１６４２に配される単一の白バイトをIOT１７０によって繰り返し読み出させる。

【００８６】IOTインターフェースコントローラ１６９１は、FIFO制御及びコマンド／マスクチャネルプロセッサ１６８０のデータ及び出力メタビットFIFO１６４２及び１６４０を充填させる部分から独立して動作する。IOTインターフェースコントローラ１６９１は、クロック信号、ページ同期信号及びライン同期信号をライン及びページ同期ハンドラー１６８８を介してIOT１７０から受信して、リターンクロック信号及び読出しデータ信号をIOT１７０へ戻す。データFIFO１６４２及び出力メタビットFIFO１６４０がマルチチャネルコンバイナ１６３０により充填された有効データを有する限り、IOTインターフェースコントローラ１６９１は、IOTによって正

確な時間でそのデータの読出しを同期するために使用される。

【００８７】図５に示されるように、バスインターフェース１６１０は、バーストFIFO１６１１、サンプルチャネル制御１６１２、スローチャネル制御１６１３、レジスタ１６１４、デコーダ１６１５、バッファーメモリ１６１６、チャネル調停ロジック１６１７及びカードコントローラ１６１８を有する。

【００８８】バスインターフェース１６１０の基本動作は、直接メモリアクセス（DMA）を使用するバス上のパケットを検索して、それらを５個のFIFOチャネル１６２０－１６２８の一つへ書込むことである。バスインターフェース１６１０は、各ページの前に１度プログラムされ、次にFIFOチャネル１６２０－１６２８を充填するために独立して動作する。チャネル調停ロジック１６１７は、FIFOFチャネル１６２０－１６２８の各々が充填される優先順位を決定するために使用される。これは、殆ど空である時に発生されるFIFO１６２０－１６２８の各々からの信号、各チャネルに対するプログラム可能”重み”及びこれらのチャネルの内の三つに対するインターリーブオプションに基づく。重みは、チャネルが現在最も高い優先順位として残る行の回数を指示する。バスインターフェース１６１０の第１の実施の形態の調停方式は、優先順位の三つのサブセットに基づく。優先順位サブセット１は、常に最も高い優先順位であり、後で詳述されるように、新たなサンプル及びマスクチャネルポインタの取出しを含む。優先順位サブセット２と３は、優先順位が交番する。優先順位サブセット２は、サンプルチャネルデータを含み、優先順位サブセット３は、ラウンドロビン取出しコマンドであり、それは、マスク、カラー、メタビット及びコマンドチャネルデータを取り出す。

【００８９】この調停方式の背景理由は、以下の通りである。ポインタ（サブセット１）は、小さく、多くないが、以下に述べられるように、重要な取出しである。サンプルチャネルデータ（サブセット２）は、システムを介して最も大きなバンド幅を要求する可能性のある生のバイトである。他のチャネルは、”スローチャネル”と呼ばれる。その理由は、それらのチャネルFIFO１６２２－１６２８が３２ビット幅よりも少ない幅を有するからである。それらを取り出すと、取り出されたワードをバーストFIFO１６１１へ書込むことが必要であり、それにより、それらは、スローチャネル制御１６１３によって並行にそれらの個別のFIFOチャネル１６２２－１６２８上にアンパックされることができる。この動作の間に、他のバースト取り出しがサンプルチャネルFIFO１６２０に対して並行に発生する可能性がある。

【００９０】各チャネルに対して取出しのバースト転送サイズを制御する事ができる。レジスタ１６１４の一つは、各チャネルに対するバーストサイズを格納する。また、各チャネルに対するデータが開始するメモリ１５０にアドレスを格納するレジスタ１６１４の一セットのレジスタがある。メモリサイズをセーブするために、マスクチャネル及びサンプルチャネルは、データ取り出しにおいて、一レベルの間接処理（indirection）を使用する。単一のアドレスポインタレジスタを使用する代わりに、マスク及びサンプルカラーチャネルの各々は、３個のレジスタを有する。第１のものは、表（テーブル）が格納されているメモリ１５０中のアドレスをポイントする。この表は、アドレス／サイズ対のリストを含み、これらの対の各々は、次のサンプルマップ又はビットマップセクションが見つけられるメモリ１５０の部分を指定する。サンプルチャネル制御１６１２は、次のアドレスをレジスタ１６１４のサンプルアドレスレジスタへ取出し、且つサンプルデータのそのブロックのそのサイズをレジスタ１６１４のサンプルサイズレジスタへ取り出す。次に、バスインターフェース１６１０は、正確な量のデータが取り出されるまで、メモリ１５０のその部分から取り出されることができ、その後、それは、サンプルデータ表中の次のアドレス／サイズ対を取り出す。マスクチャネルは、同様に処理される。

【００９１】取出しは、絶えずワード境界で発生するので、実際の第１の有効バイトのデータがワード内に発生するサンプルデータの場合に可能である。このように、サンプルチャネル制御１６１２は、取り出されようとしたメモリ１５０の中のバイトアドレスを実際に取り出されたメモリ１５０中のワードアドレスを比較して、”無効”タグで取り出された第１のワードの３バイトまでタグ付けできなければならない。サンプルチャネルアンパッカー１６３２Aは、後で、これらの無効バイトを内部サンプルFIFO１６３２Bへロードするのではなくて破棄できる。

【００９２】バスインターフェース１６１０の他の対は、データレートを等しくするためのバッファーメモリ１６１６、ローレベルSバス（Sbus）カード機能を処理するカードコントローラ１６１８、及びSバスリクエストアドレスマッピング及びデコーディングを処理するデコーダ１６１５を含む。デコーダ１６１５の一つの重要な機能は、状態情報を送信し受信できるIOT１７０のシリアルポートへのインターフェースを提供することである。

【００９３】図６は、IOTコントローラ２６０の第２の例示の実施の形態を示す。図６に示されるように、バスインターフェース１６１０と本質的な動作及び構成が同様である、バスインターフェース２６１０は、バス１１４から４バイトバーストを受信し、３２ビットデータストリームをコンプレッサ（圧縮器）／デコンプレッサ（圧縮解除器）コントローラ２６５０及び統合コンバイナ／チャネルFIFO２６３０へ出力する。統合コンバイナ／チャネルFIFO２６３０は、マルチチャネルコンバイナ１６３０をデータチャネルFIFO１６２０－１６２８と統合する。コンプレッサ／デコンプレッサコントローラ２６５０は、データをメモリ１５０からIOTコントローラ２６０へ転送する時に圧縮されたデータの使用を可能とすると共に、メモリ１５０から転送されるメタビット情報によって制御される。コンプレッサ／デコンプレッサコントローラ２６５０は、このように、メタビット情報がどのタイプの圧縮がデータの受信されたブロックへ適用されたかを指示することを可能にする。

【００９４】従って、オブジェクトタイプ（即ち、データが、”JPEG”（ジョイントフォトグラフィックエキスパートグループ）テクニックを使用して最適に圧縮されたカラー画像データか、白／黒ビットマップデータ及びラン長エンコーディング又はCCITTのような２値圧縮テクニックを使用して最適に圧縮され得る他のデータタイプであるか）に基づいてデータ圧縮を最適化できる。従って、コンプレッサ／デコンプレッサコントローラ２６５０へ入力されるデータのオブジェクトタイプに基づき、データは、サンプルデータコンプレッサ／デコンプレッサ２６２０又は２値サンプルデータコンプレッサ／デコンプレッサ２６４０へ出力される。更に、スキャナインターフェース１１６（ディジタルカメラのような他のデバイスへのインターフェースであってもよい）は、コンプレッサ／デコンプレッサコントローラ２６５０へ取り付けられることができ、それにより、リアルタイムで、スキャナからのオンザフライデータが得られ、圧縮され、コンプレッサ／デコンプレッサコントローラ２６５０へ転送され、圧縮解除され、統合コンバイナ／チャネルFIFO２６３０の適切なチャネルへ入力され得る。これにより、複写機や統合複写機／プリンタのスキャナのような、スキャナ又は他の等価の物を介して画像の部分のオンザフライ挿入を可能とし、それによりオンザフライデータは、ページ画像のPDL記述を変える必要がなく、メタビットチャネルの制御下でページ画像へ結合され得る。更に、これにより、スキャナ又は他の等価なデバイスを介して画像の部分のオンザフライ最適化を可能とする。例えば、画像の部分がより高いレベルのレーザ光を使用してレンダリングされるべきであるならば、オペレータは、増加された露光レベルセットポイントを使用して、画像のどの部分がレンダリングされるべきかを識別できる。

【００９５】サンプル又は２値データが一旦圧縮解除されると、それは統合コンバイナ／チャネルFIFO２６３０へ入力され、それは実質的に図２－４に関して上述されたように動作する。

【００９６】更に、IOTコントローラ２６０がカラー空間変換器（トランスフォーマ）２６７０を含むので、サンプルカラーデータ及び一定カラーデータは、それが格納又は転送される前に、最適なカラー空間に配されることができる。図６に示されるように、統合コンバイナ／チャネルFIFO２６３０は、２４ビットワード（カラー当り３カラーＸ８ビット）をカラー空間変換器２６７０へ出力し、この変換器は、２４ビット３カラーデータを４バイトのカラーデータへ変換し、次に、そのデータは、C、Y、M、Kカラー分離層を表す４バイトとして、統合コンバイナ／チャネルFIFO２６３０へ戻される。また、カラーデータは、バスインターフェース２６１０から直接的にカラー空間変換器２６７０へ入力され得る。どちらの場合も、メタビットデータは、カラー空間変換器２６７０を制御して現在のデータが関連する特定のオブジェクトタイプに対する最適カラー空間変換を選択するために使用され得る。オブジェクト最適化カラー空間変換の部分として、最適トーン再生カーブ（TRC）がメタビットを使用して選択され得る。次に、図２－４に関連して上述されたように動作する、統合コンバイナ／チャネルFIFO２６３０は、出力カラーデータ及びメタビットをIOT１７０へ出力する。IOT１７０の容量及びスピードに従って、最終データは、出力当り８ビットサイクルプラス１－４メタビット又は出力当り３２ビットサイクルプラス１－４メタビットで出力され得る。

【００９７】図７に示されるIOTコントローラの第３の例示の実施の形態において、コンプレッサ／デコンプレッサ３６５０は、サンプルデータコンプレッサ／デコンプレッサ３６２０を制御するためのみに使用される。図６の２値サンプルデータコンプレッサ／デコンプレッサ２６４０は、より先進の構造で置き換えされた。この例示の実施の形態において、２値コンプレッサ／デコンプレッサ３６４０は、適応コンプレッサ／デコンプレッサ及び高速CCITTコンプレッサ／デコンプレッサの何れかである。しかしながら、このIOTコントローラ３６０は、一般にIOTコントローラの第１及び第２の好適な実施の形態に関して上述されたように動作する。

【００９８】最後に、図８は、IOTコントローラ４６０の第４の例示の実施の形態を示す。この第４の例示の実施の形態において、統合コンバイナ及びチャネルFIFO４６３０は、３カラー（２４ビットワード）又は４カラー（３２ビットワード）を第１のカラー空間変換器４６７０及び第２のカラー空間変換器４６７５へ出力する。第１のカラー空間変換器４６７０及び第２のカラー空間変換器４６７５の各々は、IOT１７０へ出力される４カラー分離層の二つを発生するために使用される。一般に、カラー空間変換器の各々は、統合されたコンバイナ及びチャネルFIFO４６３０、及びパッカー４６７２と４６７７の速度の２倍の速度で動作する。

【００９９】例えば、第１のカラー空間変換器４６７０は、第１のクロックサイクルで、８ビットCカラー分離層データバイトを出力し、第２のクロックサイクルで、８ビットMカラー分離層バイトを出力する。同様に、第１のクロックサイクルで、第２のカラー空間変換器４６７５は、８ビットYカラー分離層バイトを出力し、第２

のクロックサイクルで、８ビットKカラー分離層バイトを出力する。

【０１００】第１のクロックサイクルで、第１のカラー空間変換器４６７０は、８ビットCデータをパッカー４６７２へ出力し、第２のカラー空間変換器４６７５は、８ビットYデータをパッカー４６７７へ出力する。同様に、第２のクロックサイクルで、第１のカラー空間変換器４６７０は、８ビットMデータをパッカー４６７２へ出力し、第２のカラー空間変換器４６７５は、８ビットKデータをパッカー４６７７へ出力する。パッカー４６７２と４６７７がカラー空間変換器４６７０と４６７５をの速度の半分で動作するので、次に、それらは、IOT１７０と並行に第１及び第２のカラー空間変換器４６７０と４６７５から統合された１６ビットデータを出力して全て４つのカラー分離層を４個の８ビットワードとして同時に提供する。このように、IOT１７０のより広い範囲が、このシステムで使用でき、このシステムは、IOTを介して複写シートの単一パスで、全ての４カラー分離層、C、M、Y、Kを形成するために４つの異なるプリントドラムを使用する、１パス、４カラー複写機／プリンタを含む。

【０１０１】以下の図面は、PDL分解手段１３０、IOTデータ及びコマンド命令発生手段１４０、及びメモリ１５０に格納されるデータ構造を示す。メモリ１５０は、RAM部分１５１及び不揮発性部分１５２を含む。不揮発性部分１５２は、ハードディスク、取り外し可能テープ、フロッピーディスク、光ディスク、フラッシュメモリ、及び長期間に亘る不揮発性データ記憶を提供する他の等価のメモリデバイスのいずれかより構成され得る。図９に示されるように、ステップＳ１０で開始した後、ドキュメントは、グラフィカルアーティストや他のディスクトップ出版プログラムの技術のあるオペレータのようなドキュメントクリエータによって、ステップＳ２０で作成される。

【０１０２】クリエータが一旦ステップＳ２０でのドキュメント作成を終了すると、ステップＳ３０で、IOT用のプリントデータが、オブジェクト指定又はオブジェクト最適化圧縮及びレンダリングテクニックを使用して準備される。ステップＳ３０でのプリントデータの準備は、PDL分解手段１３０及びIOTデータ及びコマンド命令発生手段１４０によって実行される。プリントデータが準備されると、それは、図２８－３１に示されるデータ構造を使用してメモリ１５０のRAM部分１５１に格納される。

【０１０３】次に、プリントデータがステップＳ３０で準備された後、ステップＳ４０において、メモリ１５０のRAM部分１５１のデータ構造がメモリ１５０の不揮発性部分１５２へ格納されるべきか否かが決定される。ドキュメントクリエータが格納ステップＳ５０を指定してもよいし、プリントシステムは、格納ステップＳ５０が

大きな又は複雑なドキュメントのリソースを保存する事を要求してもよい。メモリ１５０のRAM部分１５１に格納されるデータ構造が記憶されるべきであるということを決定すると、圧縮されたページが不揮発性メモリ部分をステップＳ５０で格納される。これは、Ｓ５０で格納されたこのドキュメントに対する圧縮されたページがプリントされるべきであることを決定した後、クリエータ又はプリントシステムが、メモリ１５０の不揮発性部分１５２からそれらをRAM部分１５１へコピーすることによって、ステップＳ６０におけるように、圧縮されたページを検索する事を必然的に要求する。このように、ステップＳ３０で準備されたプリントデータは、それがステップＳ５０で格納されてステップＳ６０で再び呼び出されたか或いはステップＳ４０によって直接送信されたかに拘らず、ステップＳ７０へ出力される。

【０１０４】また、ステップＳ２０－Ｓ５０、ステップＳ１１０－１３０、及び後で述べられるそれらのサブステップは、IOT１７０がアクティブに駆動されていない、従って、リアルタイム制約がこれらのステップを制限しないので、リアルタイムで実行される必要がないことが理解されるべきである。反対に、ステップＳ７０－Ｓ１００は、IOT１７０がアクティブに駆動されているので、リアルタイムで実行されるべきであることが理解されるべきである。従って、リアルタイムでのデータの処理及びIOT１７０への提供の故障は、IOTによってプリントされるページが不正確にされる。不揮発性メモリ部分１５２のタイプ及び容量に依存して、ステップＳ６０は、リアルタイムで実行されてもされなくてもよい。

【０１０５】ステップＳ７０において、ステップＳ３０で準備されたプリントデータは、オブジェクト指定又はオブジェクト最適化圧縮解除及びレンダリングテクニックを使用して結合及びプリントされる。同時に、ステップＳ８０で、リアルタイムデータは、捕獲されて、ステップＳ３０で準備されたプリントデータに同期される。ステップＳ８０のリアルタイムデータは、スキャナ、複写機のスキャナ部分、ディジタルカメラ、遠隔コンピュータ、或いはデータを発生してIOTコントローラ１６０へリアルタイムで送信することができる他のデバイスのような、他のソースから捕獲される。

【０１０６】次に、ステップＳ９０では、クリエータ又はプリントシステムは、このプリント動作が較正プリントであるか否かを決定する。クリエータ又はIOT１７０の自動較正処理は、ステップＳ９０で、このプリントが較正プリントであると決定すると、プリントページは、オブジェクト最適化レンダリング調節がドキュメントのオブジェクトの何れかのレンダリングを訂正することを必要とするか否かを決定するためにステップＳ１００においてリアルタイムで自動的に測定される。例えば、一つの特定のタイプの較正テストの間、IOTコントローラ１６０は、所定のテストドキュメントをIOT１７０へ出

力する。このテストドキュメントは、絵画的や写真的（サンプリングされた）オブジェクトのような、特定のタイプのオブジェクトをシミュレートするためにレンダリングされる、テストパッチを含む。IOT１７０がテストドキュメントをプリントした後、IOT１７０内の又はそれに取付られたセンサは、テストドキュメントの種々のカラーを測定して、その測定データをIOTコントローラ１６０へ提供する。IOTコントローラ１６０は、測定データによる指示の通りにプリントされた実際のカラーとオブジェクトのそのタイプに対してテストドキュメントにプリントされることが意図されるカラーを比較して調節する。このように、例えば、温度、湿度、又は他の環境ファクタにおける変化によって生じるIOT１７０のトーン再生カーブドリフトは、一定カラーオブジェクト、サンプル画像データオブジェクト又はカラーテキストオブジェクトのような、異なるオブジェクトタイプに対して、トーン再生カーブを変化することによって、オブジェクト最適化ベースで訂正され得る。このように、較正プリントが測定され且つオブジェクト最適化レンダリング調節はステップＳ１１０でなされ、制御は実際のドキュメントをプリントするためのステップＳ７０へ戻る。

【０１０７】しかしながら、ステップＳ９０で、このプリントが較正プリントでないならば、制御は、ステップＳ１１０に進み、リアルタイムではないが、そこでクリエータは、そのプリントされたドキュメントがＯＫか否かを決定する。そのプリントされたドキュメントがＯＫでない場合、制御は、ステップＳ１２０に進み、クリエータによるプリントデータ中のオブジェクトの編集が可能となり、制御はステップＳ３０へ戻る。ステップＳ３０において、平滑化（それは、一般には、オブジェクトタイプを失わせる）に先立って、オブジェクトは、オブジェクト最適化ESS１００のオペレータインターフェースを介して調節され得る。元の非分解PDLドキュメントファイルに戻る必要がなく、オペレータ又はクリエータがカラー空間変換（単数又は複数）、トーン再生カーブ（単数又は複数）及び／又は他のファクタにおける僅かな調節を出来るので、オブジェクト最適化圧縮及びレンダリングを使用してプリントデータを準備するために必要な処理の大部分が繰り返される必要がない。更に、オブジェクトタイプがこの点で保持されるので、これらのタイプの訂正が、各異なるスキャンラインに繰り返し的ではなくて、オブジェクトに対して１度なされる。

【０１０８】オブジェクト最適化ESS１００で利用できる情報を使用するこの編集ステップＳ１２０は、ドキュメントに対するレンダリング関連の変化を処理する時に、クリエータは、ワークステーションへ戻るステップ、僅かな部分のみが変化されることが必要な場合でも必要とされる全体のドキュメントをワークステーションディスプレイに再び呼び出すステップ、ドキュメントを

変更するステップ、ドキュメントの新たなPDLバージョンを生成するステップ、ドキュメントをプリンタへ再送するステップ、及び新たなプリントを見る前にドキュメントが分解処理を通過するのを待つステップの時間のかかるステップを回避する事が出来るという点で従来の技術とは異なっている。代わりに、ページ上の特定の絵画的オブジェクトのミッドトーンシアン分離に対する変更のような、ページ上の個々のオブジェクトに対するレンダリング変更がプリンタへ入力でき、そして準備プリントデータ処理ステップＳ３０の最後のサブステップのみが、クリエータが新たなプリントを見ることが出来る前に、オブジェクト最適化ESS１００内で再び実行される必要があるに過ぎない。ドキュメント生成の最終段がしばしば高度な繰り返し処理であるので、時間のセーブ量がかなり大きい。

【０１０９】更に、ワークステーションでドキュメントクリエータが利用できるレンダリング制御は、しばしばオブジェクト最適化ESS１００によって提供できる直接制御とは異なっており、且つワークステーションディスプレイはカラープリンタがプリントするカラーとは異なるようにカラーを示すので、ドキュメントクリエータは、従来の技術のワークステーションで編集することによってではなくて、編集を伴う高速再プリントステップＳ１２０を使用する事によってドキュメント内のオブジェクトに対する望ましいカラーにより素早く収束することが期待される。

【０１１０】しかしながら、プリントがＯＫであるならば、制御はステップＳ１１０からステップＳ１３０へ進み、そこでステップＳ１２０で行われた何らかの変更は、オブジェクト最適化ＥＳＳ１１０により又はドキュメント生成ソフトウエアによってドキュメントへの自動的組み込みのために標準のフォーマットにセーブされる。次に、制御は、ステップＳ１４０へ進み、そこで処理が終了する。

【０１１１】図１０は、図９のステップＳ３０のオブジェクト最適化圧縮及びレンダリングを使用してプリントデータを準備するための処理をより詳細に示す。図１０に示されるように、ステップＳ３０のプリントデータ準備処理は、ステップＳ２００で開始する。ステップＳ２００において、ステップＳ２０で生成されたドキュメントの次の（又は第１の）ページが内部PDLファイルソース手段１１０又は他の遠隔のPDLファイルソース手段１２から、現在のページとして得られる。

【０１１２】次に、ステップＳ２１０において、現在のページのオブジェクトリストが構成され、このページに対する特定オブジェクトを関連するオブジェクト最適化レンダリングタグを組込む。ステップＳ２１０において、オブジェクト最適化レンダリングタグは、各オブジェクトに対して決定されたオブジェクトタイプから、PDL分解手段１３０及び／又はIOTデータ及びコマンド命令

35

発生手段１４０によって自動的に発生されることができる。或いは、オブジェクト最適化レンダリングタグを発生するための自動処理がデフォルトモードとして使用されると共に、ステップＳ２０でドキュメントを準備する時に、ドキュメントクリエータは、PDLを使用してオブジェクトを指定する時のオブジェクト最適化レンダリングヒントを明示的に含むことが出来る。これらのクリエータ挿入レンダリングヒントは、オブジェクトタイプを明示的に定義すること、カラー空間変換又はトーン再生カーブを明示的に定義すること、ハーフトーンスクリーン周波数及び／又は角度を明示的にセットすること、そうでない場合は、PDL分解手段１３０及び／又はIOTデータ及びコマンド命令発生手段１４０によって通常自動的に設定されるガミュートマッピング及び／又は他の望ましいオブジェクト最適化レンダリングパラメータに対する優先順位を指示する最大レーザパワー変調、即ちセットポイント、を明示的にセットすること、含む。

【０１１３】オブジェクト最適化レンダリングタグを含むオブジェクトリストは、ステップＳ２１０において現在のページに対して発生されているので、ステップＳ２２０において、利用できるメモリのようなシステムリソースは、それらが破棄されなかったことを保証するためにモニタされる。例えば、ページの直交する方向に対してある角度を持って設定されたスイープを有する複雑なクリッピング領域のような、非常に複雑なページは、メモリ１５０のRAM部分１５１のメモリリソースが不充分である程度にオブジェクトリスト中に多くのオブジェクトが必要である。即ち、実際に、このページは、ナビゲート圧縮の問題を含む。

【０１１４】このように、各追加のオブジェクト毎に別のメモリを使用しないフォールバックモードが提供される。このフォールバックモードにおいて、現在のページは、サンプルチャネルにレンダリングされ、且つシステムリソースにより、低下された解像度でレンダリングされてもよい。ステップＳ２２０が、リソースが十分でないことを決定すると、制御は、ステップＳ２３０へ進み、それは、フォールバックモードを使用して現在のページに対するプリントデータを準備する。一般に、フォールバックモードは、従来のIOT１７０に使用される従来のビットマップ／バイトマップを生成することに依存する。この場合、ページは、各オブジェクト毎のプリントパラメータが最適化されるようにはプリントされないが、少なくともそのページがプリントされ得る。更に、幾つかのPDLが、前のページ（即ち、ページが独立していない）を参照し、従来のビットマップ／バイトマップに対してディフォルトすることによってリソースのオーバーフローを処理することによって現在のページに対するプリントパラメータを定義するので、ステップＳ２３０におけるように、不良ページが前に定義されたパラメータに対してそのような参照を行うことが可能である。

36

【０１１５】ステップＳ２３０において、利用可能リソースを出るページが到達すると、従来のPDL分解処理が現在のドキュメントの第１ページから開始され得る。状態及びグラフィカル演算子（オペレータ）パラメータの全てが０にリセットされる。第１ページからのグラフィカル演算子は、グラフィカル状態を正確に維持する限りのみにおいて処理される。即ち、ビットマップ又はバイトマップは、書込まれない。これは、不良ページに出会う前の最後の独立のページまで続く。その点から、グラフィカル演算子と画像データの両方が処理されるが、画像データはIOTへ出力されない。不良ページに到達すると、処理された画像データはIOTへ出力される。続くページの処理とそれらのIOTへの出力は、不良ページに出会った後の最初の独立ページまで続く。

【０１１６】この点から、オブジェクト最適化PDL分解手段１３０、及びIOTデータ及びコマンド命令発生手段１４０は、現在のドキュメントの第１ページで再スタートし、全てのデータフィールド及びグラフィカル演算子を０にリセットする。オブジェクト最適化ＥＳＳ１００は、不良ページが発見された後の最初の独立ページまで画像データを発生することなく、オブジェクト最適化処理を続ける。この点から次の不良ページまで、PDL分解手段１３０及びIOTデータ及びコマンド命令発生手段１４０は、上述のように動作する。次に、この処理は、全体のドキュメントがプリントされるまで、各不良ページ毎に繰り返される。

【０１１７】或いは、ステップＳ２３０において、二つの並行処理演算子が現在のドキュメントに対して初期化され得る。第１の処理演算子は、オブジェクト最適化処理であり得る。第２の演算子は、従来のバイトマップ／ビットマッププロセッサであり得る。オブジェクト最適化プロセッサは、第１の不良ページが発見されるまで続けることができる。この点で、第２のプロセッサは、不良ページが発見される前の最後独立ページまで画像データを発生することなく動作を開始する。この点から不良ページまで、画像データが発生されるが、IOT１７０へは出力されない。次に、不良ページ及び全ての続くページは、不良ページ後最初の独立ページまで、発生され、従来のプロセッサによってIOTへ出力される。不良ページ後の最初の独立ページに出会うと、オブジェクト最適化プロセッサは、不良ページから不良ページ後の最初の独立ページまでプリントデータをIOTへ出力することなく、グラフィカル演算子の分析を開始する。

【０１１８】この点から、オブジェクト最適化プロセッサは再びオブジェクト最適化されたデータを生成してIOT１７０へ出力し、次の欠陥ページに出会うまでそれを続ける。この時点で、従来のプロセッサのグラフィカル状態は、第２の欠陥ページの後最後の独立ページまで第１の欠陥ページの後に第１の独立ページから更新される。次に、上述の処理は、現在のドキュメントの最後の

37

ページがプリントされるまで繰り返される。

【０１１９】しかしながら、ステップＳ２２０は、メモリリソースが十分であると決定すると、制御はステップＳ２４０へ続き、そこでスキャンライン及びコマンドがスキャンラインベースで発生される。次に、ステップＳ２５０において、リアルタイムチャネルデータが抽出されてメモリ１５０に格納される。

【０１２０】ステップＳ２３０及びＳ２５０は、ステップＳ２６０へ続き、そこで現在のドキュメントの更に処理の必要なページがあるか否かを決定する。もし在るならば、制御はステップＳ２００へ戻る。しかしながら、更に処理するページがないならば、制御がステップＳ２７０へ続き、制御をステップＳ４０へ戻る。

【０１２１】図１１は、図９のステップＳ７０のオブジェクト最適化分解及びレンダリングを使用して結合し及びプリントするための処理をより詳細に示す。図１１に示されるように、ステップＳ７０の結合しプリントする処理は、ステップＳ３００で開始する。ステップＳ３００において、ページの照合されたセット（即ち、現在のドキュメントのページ）の次の（第１）ページに対する圧縮されたデータが得られる。この次のページの圧縮されたデータは、メモリ１５０のRAM部分１５１からバス１１４を介してバスインターフェース１６１０によって得られる。

【０１２２】また、圧縮されたデータは、メモリ１５０の不揮発性メモリ部分１５２へ格納され得る。これは、クリエータがステップＳ３０のオブジェクト指定圧縮及びレンダリングを使用してプリントデータを準備するための処理を繰り返す必要なしにドキュメントページのコピーを再プリントできることを望む場合に有用である。圧縮されたデータは、オブジェクトレンダリング機能を制御するためにメタビットデータを含み、それによりオブジェクト最適化能力が失われない。

【０１２３】次に、ステップＳ３１０において、現在のページの圧縮されたデータは、マルチチャネルコンバイナ１６３０を使用して圧縮解除され結合される。次に、ステップＳ３３０において、IOTコントローラ１６０へ提供されるメタビット情報は、IOT１７０によってプリントされるページを最適化するために、オブジェクト最適化レンダリング方法を選択し、同時入力ストリーム間で選択するように使用される。

【０１２４】即ち、マルチチャネルコンバイナ１６３０は、オブジェクトベースで、カラー空間変換、トーン再生カーブ、ハーフトーン発生器周波数及びスクリーン角度、最大レーザパワーセットポイント及び／又は他の情報等のパラメータを決定するためにメタビット情報を使用する事ができる。最適データは、メタビットの制御下でデータへ適用される異なるタイプの処理に基づいて、決定されてIOT１７０へ出力される。

【０１２５】また、メタビット情報の幾つかは、オブジ

38

ェクト最適化データを発生すると共に、他のメタビット情報がマルチチャネルコンバイナ１６３０によってIOT１７０へ出力されるようにマルチチャネルコンバイナ１６３０によって使用され得る。これらのメタビットは、IOT１７０のIOT１７０制御サブシステムへ出力され、且つこれらのメタビットは、プリントデータを更に最適化するために、カラー空間変換、トーン再生カーブ、最大レーザパワーセットポイント及び／又はハーフトーン発生器周波数及びスクリーン角度を含む。

【０１２６】また、メタビット情報の全ては、IOTコントローラ１６０によってプリントデータと共に、IOT１７０のサブシステムへ転送される。更に、最大レーザパワーセットポイント、カラー変換、トーン再生等のオブジェクト最適化レンダリングの幾つかの態様は、マルチチャネルコンバイナ１６３０において或いはIOT１７０において、メタビットが他の態様を制御すると共に、ステップＳ３０のオブジェクト指定レンダリング及び圧縮を使用してプリントデータを準備するための処理の間に適用され得る。

【０１２７】現在のページが、ステップＳ３３０において、IOT１７０によってオブジェクト最適化フォームでプリントされると、制御は、ステップＳ３４０へ進み、そこで現在のコピーの最後のページがプリントされたか否かを決定する。最後のページがまだプリントされていない場合、制御はステップＳ３００へ戻る。しかしながら、現在のコピーの最後のページのプリントが完了すると、制御はステップＳ３５０へ続き、そこで希望のページの全てがプリントされたか否かが決定される。もしそうでないならば、制御はステップＳ３６０へ進み、そこでコピーの数が１だけインクリメントされ、次にステップＳ３７０へ進み、現在のページのポインタが現在のドキュメントの最初のページへリセットされ、最後に、ステップＳ３００へ戻る。しかしながら、ステップＳ３５０が最後のページがプリントされたと決定すると、制御はステップＳ３８０へ続き、制御をステップＳ９０へ戻す。

【０１２８】図１２は、図１０のステップＳ２１０のオブジェクト最適化レンダリングタグを有するオブジェクトリストを構成するための処理をより詳細に示す。図１２に示されるように、オブジェクトリストの構成は、ステップＳ４００においてPDLドキュメントファイルを読み出すことによって開始する。次に、ステップＳ４１０において、次の（最初）の言語要素が現在の言語要素として得られ、解析される。現在の言語要素が解析された後、標準のグラフィックスインターフェースが呼び出される。標準のグラフィックスインターフェースに導くPDL及び他の動作の解析は、ひとつのPDLから他のPDLに変化する。これらの処理は公知である。

【０１２９】次に、ステップＳ４２０において、現在の言語要素は、それが現在のページの終わりを指示するか

否かを決定するために分析される。現在の言語要素が現在のページの終わりを指示するならば、制御はステップＳ４３０へ進み、現在のページのクリッパー領域が検証される。クリッパー検証処理は、図１９のステップＳ１１４０－Ｓ１１６０に関連して更に詳細に述べられる。現在のクリッパー領域がステップＳ４３０において検証されると、制御はステップＳ４４０へ進み、制御がステップＳ２２０へ戻される。

【０１３０】現在の言語要素がステップＳ４２０で現在のページの終わりを指示しないならば、制御がステップＳ４５０へ進む。ステップＳ４５０において、現在の言語要素は、それがカラー演算子であるか否かを知るためにチェックされる。もしそうなら、制御がステップＳ４６０へ進み、カラー演算子処理が実行される。

【０１３１】しかしながら、ステップＳ４５０において、現在の言語要素がカラー演算子でないことが決定されると、制御がステップＳ４７０へ進み、現在の言語要素は、それがマスキング演算子であるか否かを知るためにチェックされる。もしそうならば、制御がステップＳ４８０へ進み、マスキング演算子処理が実行される。

【０１３２】しかしながら、ステップＳ４７０において、現在の言語要素がマスキング演算子でないならば、制御がステップＳ４９０へ進み、次の言語要素は、それが状態演算子であるか否かを知るためにチェックされる。もしそうならば、制御がステップＳ５００へ進み、グラフィカル状態演算子処理を実行する。ステップＳ４６０、Ｓ４８９及びＳ５００の全てがステップＳ４１０へ戻る。

【０１３３】最後に、現在の言語要素がステップＳ４９０において状態演算子でないことが決定されると、制御がステップＳ５１０へ進み、エラー指示がオブジェクト最適化ＥＳＳ１００によって現在の言語が適切に解析され得ないことを指示するために出力される。次に、ステップＳ５１０から制御がステップＳ５２０を介してステップＳ２０へ戻る。

【０１３４】ステップＳ４５０でチェックされるカラー演算子が、"setcolor"、"image"、"colorimage"、及び等価のコマンドのような演算子を含むことが理解されるべきである。勿論、実際のコマンドは、PDL言語に依存する。同様に、ステップＳ４７０でチェックされるマスキング演算子コマンドは、"fill"、"stroke"，"character"及び他のこのようなコマンドを含む。また、実際のコマンドは、PDL言語に依存する。最後に、ステップＳ４９０でチェックされる状態演算子コマンドは、"setclipper"、"setstrokewidth"、"setrenderhint"、及び他のそのような演算子のようなコマンドを含む。また、実際のコマンドが使用されるPDL言語に依存する事が理解されるべきである。

【０１３５】図１３は、図１０のステップＳ２４０のスキャンラインデータを発生するための処理をより詳細に示す。図１３に示されるように、スキャンラインデータを発生するための処理ステップＳ６００でアクティブオブジェクトリストを初期化しスキャンラインを１にセットすることによって開始する。

【０１３６】次に、ステップＳ６１０において、現在のスキャンラインからの各新たなオブジェクトが整列されたアクティブオブジェクトリストへ併合される。図２８は、ステップＳ２１０においてオブジェクト指定タグを有するオブジェクトリストの構成の間発生されるスキャンラインオブジェクトリストデータの汎用フォームを示す。図２８に示されるように、第３のスキャンライン１５１３上でスタートするオブジェクトを指すポインタは、例えば、第３のスキャンライン１５１３上の第１の（最初の）オブジェクト１５１３１を指し、それは、一方、スキャンライン１５１３上の第２のオブジェクト１５１３２を指す。この構造において、「１５１３１」、「１５１３２」のようなオブジェクトリストは、それらを生成する対応するPDL言語要素の相対順位に従ってリストに位置決めされるオブジェクトを含む。

【０１３７】しかしながら、整列された（ソートされた）アクティブオブジェクトは、オブジェクトの開始位置により且つ左から右の順序で整列される。このように、整列されたアクティブオブジェクトリストの第１のオブジェクトは、最左開始位置を有するアクティブオブジェクトである。同様に、各スキャンライン上の最後のオブジェクトは、最右開始位置を有するを有するオブジェクトである。所与のスキャンラインを処理すると、そのスキャンライン始まる各オブジェクトは、この左から右への順序付けが維持されるように、アクティブリストに併合される。一般に、アクティブオブジェクトリストは、現在のスキャンラインに先立つスキャンラインで始まり且つアクティブのままであるオブジェクトを含む。

【０１３８】ステップＳ６１０で新たなオブジェクトが整列されたアクティブオブジェクトリストに併合されると、制御がステップＳ６２０へ進む。ステップＳ６２０において、現在のスキャンラインに対するランリストが整列されたアクティブオブジェクトリストから発生される。次に、制御がステップＳ６３０へ進み、そこで現在のスキャンラインに対するランリストが、スキャンラインの各点に沿って、ランリストのどのランが"頂部"に在るかを決定することによって、即ち、その点であらゆる他のランの下に無いランを決定することによって、平滑化される（flattened）。

【０１３９】ステップＳ６３０において、ランリストが平滑化されると、制御がステップＳ６４０へ進む。ステップＳ６４０において、現在のスキャンラインに対するコマンド及びカラーの順序付けリストが生成される。コマンドは、適切なカラーがカラーレジスタで利用できること、マスクデータが適切にイネーブル又はディスイーブルされること、サンプリングされた画像データが必要に応じて利用できること、メタビットが、圧縮解除、カラー空間変換、トーン再生カーブ処理、IOTコントローラ１６０やIOT１７０によって実行されるレーザ変調セットポイント等のハードウェア処理を最適化するように適切にセットアップされること、を確実にするために発生される。パレット参照のフォームでのカラーは、正確なカラー及びレンダリングタグ情報が各命令に対してそのパレットから読み出されることを確実にするために発生される。

【０１４０】ステップＳ６４０の後に、制御がステップＳ６５０へ進み、そこで消費されたアイテムが整列されたアクティブオブジェクトリストから除去される。現在のスキャンラインはオブジェクトが表れる最後のスキャンラインである時に、オブジェクトが消費される。次に、ステップＳ６６０において、スキャンライン数は、現在のスキャンライン数がページの最後のスキャンラインであるか否かを決定するためにチェックされる。もしそうでないならば、制御がステップＳ６７０へ進み、そこでスキャンラインの数が１だけインクリメントされる。ステップＳ６７０から、制御フローがステップＳ６１０へ戻る。しかしながら、現在のスキャンラインが最後のスキャンラインであるならば、制御はステップＳ６９０を介してステップＳ２５０へ戻される。

【０１４１】ステップＳ２４０でスキャンラインデータが発生された後、制御がステップＳ２５０へ進む。図１４は、ステップＳ２５０をより詳細に示す。処理は、コマンドデータを抽出、圧縮及び格納する事によってステップＳ７００で開始する。コマンドデータの特徴のために、レンペルージプーウェルチ（LZW）のような公知の圧縮技術がコマンドデータを圧縮するために使用される。コマンドデータは、平滑化ランからスキャンラインベースで抽出される。コマンドデータは、図３１に示されるように、RAM部分１５１のコマンドチャネルデータ部分に格納される。

【０１４２】次に、ステップＳ７１０において、一定カラーデータが、以下で詳述されるように、抽出、圧縮及び格納される。次に、ステップＳ７２０において、メタビットデータが抽出、圧縮及び格納される。ステップＳ７３０において、マスクポインタが抽出及び格納され、それらのマスクポインタが指すマスクデータが抽出、圧縮及び格納される。コマンドデータの場合のように、LZWのような従来の技術が一定カラー及びメタビットデータを圧縮するために使用される。上述のステップの場合のように、一定カラーデータ、メタビットデータ及びマスクポインタは、平滑化ランリストから抽出され、スキャンラインベースで、夫々RAM部分１５１の一定カラーチャネル部分、メタビットチャネル部分及びマスクデータチャネル部分へ格納される。このマスクデータは、同様に抽出されて、次にラン長エンコーディング、CCITTGroup４及び他の互換システムのような既知の１ビット圧縮技術を使用して圧縮される。一旦圧縮されると、マスクデータは、またRAM部分１５１のマスクデータチャネル部分に格納される。

【０１４３】次に、ステップＳ７４０において、サンプリングされた画像ポインタが抽出及び格納され、且つサンプリングされた画像データが抽出、圧縮及びRAM部分１５１のサンプルデータチャネル部分に格納される。ステップＳ７１０－Ｓ７４０の処理は、非常に類似している。しかしながら、国際標準化機構（InternationalStandards Organization）のJoint Photographic Expert Group（JPEG）によって定義されたもののような、データのタイプ毎に異なる圧縮技術が使用される。ステップＳ７１０－Ｓ７４０で異なるデータ部分及びタイプの全てが一旦抽出、（恐らく）圧縮及び格納されると、制御がステップＳ７５０を介してステップＳ２６０へ戻る。

【０１４４】図１５は、図１２のステップＳ４６０の、カラー演算子を処理する方法をより詳細に示す。カラー演算子処理は、ステップＳ８００でカラー演算子がサンプリングされた画像を提供するか否かを決定することによって、開始する。もしそれがサンプリングされた画像でないならば、制御がステップＳ８１０へ進む。ステップＳ８１０において、カラーパレットは、現在の言語要素によって指示される一定カラーがカラーパレットに同一のエントリをすでに有するか否かを決定するためにチェックされる。パレットエントリが最初に確立されると、そのレンダリングタグフィールドがデフォルト値にセットされる。そのレンダリングタグがデフォルト値を残す限り、同一のパレットエントリがあるか否かを知るためにチェックすると、答えは肯定である。もしそうならば、制御がステップＳ８２０へ進み、そこで現在の次の言語要素によって指示されるのと同じカラーモデル及び画素データを有するパレットエントリが検出される。

【０１４５】しかしながら、例えば、図１６のステップＳ１０５０によって、レンダリングタグが変更されているならば、次に、答えは否定であり、新たなパレットエントリが生成される必要がある。このように、ステップＳ８１０の答えが否定であると、制御がステップＳ８３０へ進み、其処で新たなパレットエントリが、現在の言語要素によって指示されるカラーモデル及び画素データを使用して、カラーパレット内に確立される。次に、ステップＳ８４０において、ハッシング関数（機能）が、適切なパレットインデックスを決定するために、現在の言語要素によって指示される画素データに適用される。図３０に示されるように、パレットは、複数のインデックススロット１５２１－１５２ｎよりなる。パレット１５２０の各ハッシュ表（テーブル）は、多くのパレットエントリを有し、パレットエントリの各々は、一定カラーエントリ及びサンプリングされたエントリの何れかであり得る。パレットは、後でより詳細に述べられる。次

に、ステップＳ８５０において、現在のパレットエントリが、パレットインデックスによって決定されるハッシュテーブルスロットでパレットに挿入される。

【０１４６】新たなパレットエントリのセットアップの代わりに、現在の一定カラーは、ステップＳ１０５０がこの現在の一定カラーに対してレンダリングタグをセットする事を必要とするまで、別に格納できる。この場合、この時点でのみ、新たなパレットエントリがデフォルト値からすでにリセットされているレンダリングタグフィールドで形成される。

【０１４７】しかしながら、ステップＳ８００がカラー演算子がサンプル画像を提供した事を決定すると、制御がステップＳ８６０へ進む。ステップＳ８６０で、画像解像度除数が画像データ及び利用できるIOT解像度に基づいて計算される。表４のビット６及び７を参照して述べられたように、画像解像度除数は、IOTに対して得ることが出来る解像度を発生するために、サンプリングされた画像の各画素が何クロックサイクル繰り返されるかを指示する。

【０１４８】次に、ステップＳ８７０において、現在の画像データは、ステップＳ８６０で決定された画像解像度除数及びグラフィカル状態演算子処理によって評価された最近の”現在変換（currenttransform）”に従って、回転及び／又はスケーリングされる。次に、ステップＳ８８０において、新たなカラーパレットエントリは、現在のカラーモデル及び現在のサンプリングされた画像データを使用して、確立される。

【０１４９】次に、ステップＳ８９０において、パレットインデックスは、ハッシング関数を画像データハイト即ち”S－Size（Sサイズ）”へ適用することによって計算される。次に、ステップＳ９００において、現在のパレットエントリは、パレットインデックスによって決定されたハッシングテーブルスロットでカラーパレットに挿入される。

【０１５０】次に、ステップＳ８２０、Ｓ８５０及びＳ９００の何れかから、制御がステップＳ９１０へ進み、そこでグラフィカル状態演算子処理で発生された”currentcolor（現在のカラー）”ポインタが、ステップＳ８２０、Ｓ８５０又はＳ９００で決定された現在のパレットエントリにセットされる。次に、制御がステップＳ９２０を介してステップＳ４１０へ進む。

【０１５１】図１６は、図１２のステップＳ４８０のマスキング演算子を処理する方法をより詳細に示す。マスキング演算子処理は、パラメータ”renderhint（レンダリングヒント）”がグラフィカル状態演算子処理によってすでにセットされているか否かを決定することによって、ステップＳ１０００で開始する。もしそうでないならば、制御がステップＳ１０１０に進み、そこでオブジェクト最適化レンダリングタグが自動的に決定される。しかしながら、パラメータ”renderhint”がセットされると、制御がステップＳ１０２０へ進み、そこでオブジェクト最適化レンダリングタグが”renderhint”パラメータ（単数又は複数）から導出される。

【０１５２】即ち、パラメータ”renderhint”がセットされないならば、オブジェクト最適化レンダリングタグは、現在の言語要素の決定されたオブジェクトタイプ及びオブジェクトの異なるタイプ間で提供される区別のレベルの数に依存して、現在の言語要素に対してオブジェクトタイプを最初に分析することによって決定される。区別の単一レベルのみがイネーブルならば、ステップＳ１０１０において決定されたオブジェクト最適化レンダリングタグは、例えば、非画像オブジェクト（例えば、テキストやラインアートオブジェクト）から画像オブジェクト（例えば、ハーフトーンオブジェクト）を区別する。もし区別の追加のレベルがイネーブルであったならば、追加のレンダリングタグは、同じオブジェクトに異なるカラー空間変換、異なるトーン再生カーブ、異なる飽和レベル等が提供されるように、発生される。何れの場合も、少なくとも一つのレンダリングタグは、そのオブジェクトのデータを書込む時に使用されるレーザパワーセットポイントを指示するために各オブジェクト毎に発生される。

【０１５３】”renderhint”パラメータがセットされると、ステップＳ１０２０において、自動的に決定された値、そうでなければステップＳ１０１０で決定された値は、ドキュメントクリエータから明示的命令によってオーバーライドされる。このように、ドキュメントクリエータは、そうでなければステップＳ１０１０でセットされたレンダリングタグをオーバーライドできる。従って、”renderhint”パラメータは、必要でないかもしれないが、オブジェクトタイプが何であるか指定できる。それは、このオブジェクトタイプがどのオブジェクトタイプであるかに拘らず、ドキュメントクリエータが観察者の目を獲得したいと思う物を指示するヒントの独立のタイプであっても良い。或いは、”renderhint”は、このオブジェクトが背景にあり、観察者の目を獲得したくないことを指示しても良い。”renderhint”は、オブジェクトのエッジがシャープにされるべきでない事を指示してもよい。”renderhint”は、このオブジェクトの定義されたカラーが飽和がブーストされるのではなくて保持されるべきであることを指示してもよい。”renderhint”は、オブジェクトのそのタイプに対してデフォルトセットポイントとは異なる特定のレーザパワーセットポイントが代わりに使用されるべきであることを指示してもよい。

【０１５４】また、”renderhint”は、定義されたオブジェクトタイプ及び未定義のサブパラメータに基づいて、デフォルトモードに続く分析の残りの部分でオブジェクトタイプを明示的に定義してもよい。

【０１５５】次に、ステップＳ１０１０及びステップＳ１０２０から、制御がステップＳ１０３０へ進み、其処でグラフィカル状態演算子処理によってセットされた”currentcolor”状態のレンダリングタグが更新される。図１２，１５及び１６に示されるように、”currentcolor”タグがデフォルトタグでなく、”currentcolor”によって指示されたパレットエントリに対するレンダリングタグと一致しないならば、そのパレットエントリは、複写されなければならず、その新たなパレットエントリのレンダリングタグが更新され、”currentcolor”ポインタがその新たなパレットエントリに更新される。

【０１５６】また、その新たなパレットエントリが処理カラー演算子ステップＳ４６０の間に生成されない場合、カラーデータは、パレットへ挿入されるのではなくて、別途保持される。この場合、新たなパレットエントリは、ステップＳ１０３０で生成され、そのカラー値は、別途保持された所にセットされる。この新たなパレットエントリのレンダリングタグは、ステップＳ１０１０又はステップＳ１０２０において決定された値にセットされる。この新たなパレットエントリは、ステップＳ１０５０の部分としてステップＳ８４０、Ｓ８５０及びＳ９１０を実行する事によって、この時にパレットへ入力される。

【０１５７】次に、ステップＳ１０４０において、現在のオブジェクトは、ボックス、ビットマップ等の一つ以上のプリミティブ（主）オブジェクトを発生するためにスキャン変換される。スキャン変換は公知の処理である。次に、ステップＳ１０５０で、プリミティブオブジェクトの全てが処理されたか否かが決定される。いずれかのプリミティブオブジェクトが残っていると、制御がステップＳ１０５０からステップＳ１０６０へ進み、其処で次の（最初の）プリミティブオブジェクトが現在のプリミティブオブジェクトとしてマスキング演算子から得られる。次に、ステップＳ１０７０において、プリミティブマスキングオブジェクト処理が現在のプリミティブオブジェクトに対して実行される。次に、制御は、ステップＳ１０５０へ戻る。このループは、ステップＳ１０５０が処理されるべきプリミティブオブジェクトがないことを決定するまで続く。この場合、制御がステップＳ１０８０を介してステップＳ４１０へ戻る。

【０１５８】図１２のステップＳ５００で述べられたグラフィカル状態演算子処理のより詳細なバージョンを示す。図１７において、ステップＳ５００のグラフィカル状態演算子処理は、”setclipper”演算子がセットされたか否かをステップＳ１１００で決定する事により開始する。

【０１５９】”setclipper”演算子がセットされていないならば、制御がステップＳ１１１０に進み、”setrenderhint”演算子がセットされたか否かを決定する。もしそうでないならば、制御がステップＳ１１２０に進み、この状態演算子に対してグラフィカル状態をセットし、ステップＳ１１８０に進む。しかしながら、”setrenderhint”演算子がステップＳ１１１０でセットされると、制御がステップＳ１１３０に進み、そこでグラフィカル状態における”renderhint”パラメータは、”setrenderhint”演算子によって指示されるレンダリングヒントにセットされる。ステップＳ１１３０から制御が再びステップＳ１１８０進む。

【０１６０】しかしながら、ステップＳ１１００において、”setclipper”演算子がセットされると、制御がステップＳ１１４０へ進み、そこで現在のクリッパーオブジェクトが存在し、その完全性属性が”完全な”であるか否かが決定される。もしそうならば、制御がステップＳ１１５０へ進み、そこで現在の完全なクリッパーオブジェクトがスイープオブジェクトへ変換される。次に、領域ｒ１として示されるスイープアウトラインが、現在のクリッピング領域にセットされる。同時に、領域ｒ２で示されるスイープのフィル（充填）領域が現在のクリッピング領域の下のスイープオブジェクトにセットされる。しかしながら、現在のクリッパーオブジェクトが存在しない場合又は現在の既存のクリッパーオブジェクトが”complete（完全な）”の完全性属性を持たない場合、制御がステップＳ１１４０から直接にステップＳ１１６０へ進む。

【０１６１】ステップＳ１１６０において、このオブジェクトが、オブジェクトの最初のスキャンラインとして、スキャンライン整列アクティブオブジェクトリストに挿入される。ステップＳ１１４０からステップＳ１１６０は、図１２に示され且つ上述された”クリッパー検証”処理Ｓ４３０を形成する。次に、制御がステップＳ１１７０に進み、そこでグラフィカル領域の”currentclipper”コマンドが新たなクリッパー領域にセットされる。次に、ステップＳ１１２０及びステップＳ１１３０において、制御がステップＳ１１８０へ進み、制御がステップＳ４１０へ戻る。

【０１６２】図１３のステップＳ６２０において述べられたように、スキャンラインリストを発生するための処理のより詳細な記述が図１８において示される。スキャンラインリストを発生するための処理は、変数”thisobject”を現在のスキャンラインのアクティブオブジェクトリストの最初のオブジェクトへセットすることにより、ステップＳ１２００において開始する。

【０１６３】次に、ステップＳ１２１０において、変数”thisobject”は、それが有効オブジェクトを指すか否かを決定するためにテストされる。もし変数”thisobject”が有効オブジェクトを指さないならば、制御がステップＳ１２８０を介してステップＳ６３０に戻る。しかしながら、”thisobject”が有効オブジェクトを指すならば、”thisobject”が最初のオブジェクトにセットされた直後に常に真となり、制御がステップＳ１２２０に進む。

【０１６４】ステップＳ１２２０において、" thisobject" は、それがスイープタイプのオブジェクトであるか否かを知るためにチェックされる。もしそうならば、制御がステップＳ１２３０に進む。ステップＳ１２３０において、現在のスキャンライン上の" thisobject" のｓ１アウトラインの各ピース毎に一つのランが整列されたランリストに併合される。併合されたランの各々は、そのスキャンラインに沿った開始位置及び終了位置からなる。各ランは、また" thisobject" の下層のスイープオブジェクトｓ２を指すポインタからなり、層及びカラーデータが抽出され得る。上述のように、ｓ１領域は、スイープが生成された時に有効にクリッピング領域へセットされ、他方ｓ２領域は現在のクリッピング領域の下層のスイープオブジェクトへセットされた。ステップＳ１２３０を実行した後、制御がステップＳ１２７０へ進む。

【０１６５】ステップＳ１２２０において、" thisobject" のクラスがスイープでない時、制御がステップＳ１２４０へ進み、そこで" thisobject" は、それがクリッパータイプオブジェクトであるか否かを知るためにチェックされる。もしそうならば、制御がステップＳ１２５０に進む。ステップＳ１２５０において、" thisobject" インサイドリストから各オブジェクトが" thisobject" のクリッパーに抗してクリップされる。インサイドリストは、図２３において示され且つ以下でより詳細に述べられる、ステップＳ１７４０－Ｓ１７７０のプリミティブマスキングオブジェクト処理の間に、このクリッピングオブジェクトのために、収集されると共にこのリストへ追加されたオブジェクトのリストである。即ち、オブジェクトの各インサイドリストは、現在のクリッピング領域として対応するクリッピングオブジェクトが有効であった間対応するクリッピングオブジェクトに関連する。" thisobject" のクリッパーは、ｓ１スイープアウトラインに関して上述されたの同様に、このクリッパーオブジェクト生成された時に有効であったクリッピング領域へセットされた。オブジェクトは、クリッピング領域のクリッパーの外側にあるオブジェクトのあらゆる部分を除去する事によってクリップされる。

【０１６６】オブジェクトがクリップされた後、各得られたランは、整列されたランリストに併合される。ステップＳ１２２０において、各ランはスキャンラインに沿う開始位置と終了位置からなる。しかしながら、ステップＳ１２５０において、また各ランは、" thisobject" のインサイドリストから層及びカラーデータへのクリップされたオブジェクトに対するポインタからなる。ステップＳ１２３０における場合のように、処理が一旦完了すると、制御がステップＳ１２７０へ進む。

【０１６７】もしステップＳ１２４０において、" thisobject" のクラスがクリッパーでない場合、制御がステップＳ１２６０に進む。ステップＳ１２６０において、現在のスキャンラインの" thisobject" の各ピース毎のランが整列されたランリストへ併合される。ステップＳ１２３０及びステップＳ１２５０の場合のように、各ランはスキャンラインに沿う開始位置及び終了位置からなる。しかしながら、この場合、各ランは、また層及びカラーデータに対する" thisobject" へのポインタからなる。ステップＳ１２３０及びステップＳ１２５０の場合のように、一旦ステップＳ１２６０の処理が終了すると、制御がステップＳ１２７０へ進む。

【０１６８】ステップＳ１２７０において、変数" thisobject" が現在の" thisobject" の" 次の" フィールドによって指示されるオブジェクトへセットされる。従って、次の" thisobject" が現在の" thisobject" になる。ステップＳ１２７０から制御がステップＳ１２１０へ進む。

【０１６９】図１９は、図１３のステップＳ３６０のランリストを平滑化するための処理をより詳細に示す。図１９に示されるように、ランリストを平滑化する処理は、ステップＳ１３００で開始する。ステップＳ１３００において、現在のランの前景及び背景オブジェクトは、デフォルトレンダリングタグを有する白オブジェクトに初期化される。開始位置及び終了位置は、現在のスキャンラインの開始に初期化される。最後に、変数" currentstart" がスキャンラインの開始に初期化される。

【０１７０】次に、ステップＳ１３１０において、変数" currentstart" は、スキャンラインの終了に達したか否かを決定するためにチェックされる。そうでない場合、制御がステップＳ１３２０に進み、そこで次の可視ランが識別される。

【０１７１】次に、制御がステップＳ１３３０に進み、そこで次の可視ランの前景及び背景オブジェクトカラーとレンダリングタグは、それらが現在のランのオブジェクトの前景及び背景オブジェクトカラーとレンダリングタグと同じであるか否かを決定するためにチェックされる。これが真ならば、制御がステップＳ１３６０に進み、そこでランは、現在のランの終了位置を次の可視ランの終了位置へセットする事によって、結合される。しかしながら、これが真でない場合、制御がステップＳ１３４０へ進み、そこで現在のランに対するコマンド及びカラーが生成される。次に、ステップＳ１３５０において、次の可視ランが現在のランになる。

【０１７２】ステップＳ１３５０及びＳ１３６０の両方がステップＳ１３７０に進み、そこで" currentstart" が現在のランの終了位置にセットされる。次に、ステップＳ１３７０からの制御がステップＳ１３１０に進む。ステップＳ１３１０において、" currentstart" がスキャンラインの終了へ到達されると、制御がステップＳ１３８０に進み、そこでコマンドおよびカラーの最後のセットが現在のランのために生成される。ステップＳ１３８０から、制御がステップＳ１３９０を介してステップＳ６４０へ戻る。

【０１７３】図２０は、図１９のステップＳ１３４０またはＳ１３８０の現在のスキャンラインの現在のランに対するコマンド及びカラーを生成するための処理をより詳細に示す。ステップＳ１４００において、現在のランの前景オブジェクトは、そのオブジェクトが透明か否かを決定するために調べられる。即ち、その照会は、そのオブジェクトが真のクリアフィールド１５５７４を有するビットマップタイプのオブジェクト１５５７０であるか否かということである。もしそうでないならば、制御はステップＳ１４１０へ進む。ステップＳ１４１０において、現在のランの前景オブジェクトは、そのオブジェクトがクラススイープタイプオブジェクト１５５８０であるか否かを知るために調べられる。もしそうでないならば、制御がステップＳ１４２０に進み、ノーマルコマンド及びカラーを生成する。次に、制御がステップＳ１４６０に進む。

【０１７４】もしステップＳ１４１０において、前景オブジェクトがスイープクラスタイプオブジェクト１５５９０であると決定されると、制御がステップＳ１４３０に進み、スイープコマンド及びカラーを生成する。次に、制御が再びステップＳ１４６０に進む。しかしながら、ステップＳ１４００において、現在のランの前景オブジェクトが真のクリアフィールドを有するクラスビットマップタイプのオブジェクトであると決定されと、制御がステップＳ１４４０に進み、背景チェーンを処理してマスクビットマップを固定する。次に、制御がステップＳ１４５０に進み、マスクを使用してコマンド及びカラーを生成する。次に、制御が再びステップＳ１４６０に進む。

【０１７５】この手順は、それが２つのステップに依って呼び出され得る点において一般的ではない。このように、ステップＳ１４６０は、ステップＳ１３４０及びＳ１３８０のどちらがこの手順を呼び出したかを決定して、制御が適切な次のステップ、即ちステップＳ１３５０又はＳ１３７０、へ戻される。即ち、現在のラン手順に対してコマンド及びカラーの生成がステップＳ１３４０で呼び出されると、制御がステップＳ１３５０に進む。同様に、現在のラン手順に対してコマンド及びカラーの生成がステップＳ１３８０で呼び出されると、制御がステップＳ１３９０に進み、制御をステップＳ６４０へ戻す。

【０１７６】上述のように、ステップＳ１４００は、現在のランの前景オブジェクトが透明（即ち、ビットマップクラスオブジェクト）の場合、その背景フィールドによって指示されるオブジェクトがこのランのマスクの" 0" ビットのカラーを決定すると共に、現在のランのオブジェクトが前景カラーを決定するように動作する。しかしながら、背景オブジェクトそれ自体は、同じ又は異なる前景カラーを有する透明ビットマップであってもよい。前景カラーが同じ場合、黒文字ビットマップを有する場合と同様に、二つの隣接する透明層は、論理ＯＲ関数をそれらのビットマップに適用することによって結合され得る。この領域のマスクチャネルマップの元の書込みが下のビットに上書きされたので、下のビットマップオブジェクトのビットマップがこのステップでそのマップにＯＲされる。透明背景をその上のオブジェクトにＯＲして結合する事は、上のオブジェクトとは異なる" 1" ビットのカラーを使用して透明背景層に到達されるまで続くことができる。この点で、マスクチャネルの望ましい使用におけるコンフリクトが見られた。それは、コンフリクトする透明背景ビットマップの空間的に隣接する" 1" ビットのグループをランに変換して" 平滑化" 処理を再帰的に適用することによって解決される。このように、コンフリクトする透明背景オブジェクトが透明でないオブジェクトのセットに変換される。この処理は、不透明オブジェクト又は層" 0" 白ペーパが発見されるまで、背景チェーンに続く。

【０１７７】図２１は、図１４のステップＳ７１０の一定カラー抽出、圧縮及び格納処理のより詳細な記述を示す。抽出、圧縮及び格納一定カラー処理はステップＳ１５００で開始する。ステップＳ１５００において、次の（最初の）カラー参照（基準）が現在のカラー参照として得られる。現在のカラー参照は、図３０に示されるパレットデータ構造のパレットエントリの一つを指すポインタである。

【０１７８】次に、ステップＳ１５１０において、現在のパレットエントリからカラーデータが得られる。図３０に示されるように、各一定カラータイプパレットエントリ１５２３０は、このハッシュテーブルスロットの次のパレットエントリを指す次のリンク１５２３１と、このパレットエントリがどのようにレンダリングされるべきかについてのデータを提供するレンダリングタグフィールド１５２３２、及びこれが一定カラーパレットエントリであるか、サンプリングされた画像パレットエントリであるか、他のタイプのパレットエントリであるかを指示するカラークラスフィールド１５２３３を含む。カラーモデルフィールド１５２３４は、画素データがどのタイプのカラーモデル（例えば、RGB、CMYK、CIELab、又はその他）を使用するかを指示すると共に、レンダリングタグフィールドと共にこのカラーをCMYKに変換するための好ましいカラー空間変換を指示する。最後に、画素値フィールド１５２３５は、実際のカラーデータを格納する。

【０１７９】次に、ステップＳ１５２０において、現在のパレットエントリの画素データは、画素値データフィールド１５２３５から読み出される。次に、ステップＳ１５３０において、丁度読み出された画素データが、図３１に示されるように、一定カラーチャネルの次に利用できる位置でRAM１５１のチャネルデータ構造に格納

される。

【０１８０】カラーモデルの混合物からなる画素データがこのようにチャネルデータ構造に格納されることが予知される。それは、上述のように、IOT１７０又はマルチチャネルコンバイナ１６３０においてメタビットの制御下でCMYK（又は他のIOT指定）データに変換される。また、カラーモデル及びレンダリングタグフィールドによって決定されたカラー変換は、画素データに適用でき、ステップＳ１５２０の一部としてそれをCMYKに変換できる。

【０１８１】次に、ステップＳ１５４０において、パレットは、現在のカラー参照が最後のカラー参照であるか否かを決定するためにチェックされる。もしそれがそうでないならば、制御がステップＳ１５００に戻り、その次のカラー参照が再び現在のカラー参照として選択される。しかしながら、現在のカラー参照が最後のカラー参照である場合、制御がステップＳ１５５０に進む。

【０１８２】ステップＳ１５５０において、一定カラーチャネルデータチャネル構造におけるカラーデータは、メモリから読み出され、圧縮されて、圧縮フォームで再格納される。他のチャネルデータの場合のように、一定カラーチャネルフィールドに格納される一定カラーデータを圧縮する事によって、バス１１４を介するデータ転送が最小化される。

【０１８３】一定カラーデータが圧縮されてステップＳ１５５０に格納されると、制御がステップＳ１５６０を介してステップＳ７２０に戻る。

【０１８４】図２２は、図１４のステップＳ７２０のメタビット抽出、圧縮及び格納処理のより詳細な記述を示す。図２２に示されるように、メタビットを抽出、圧縮及び格納する処理は、ステップＳ１６００で開始する。

【０１８５】図２１のステップＳ１５００におけるのと同様に、ステップＳ１６００において、次の（最初の）カラー参照（それはパレットエントリに対するポインタである）が現在のカラー参照として得られる。同様に、ステップＳ１６１０におけるのと同様に、ステップＳ１６１０において、レンダリングタグは、一定カラーパレットエントリ１５２３０のレンダリングタグフィールド１５２３２及びサンプリングされた画像パレットエントリ１５２４０のレンダリングタグフィールド１５２４２から得られる。両方の一定カラーパレットエントリ１５２３０及びサンプリングされた画像パレットエントリ１５２４０は、共にレンダリングタグを含むので、この処理は、図２１に示されるフローチャートにおけるように、一定カラーパレットエントリに限定されない。

【０１８６】ステップＳ１６１０でレンダリングタグが得られると、制御がステップＳ１６２０に進む。ステップＳ１６２０において、レンダリングタグは、プリンタ独立レンダリングタグとIOT指定メタビットとの間で指定の変換を行う変換テーブル中でルックアップされる。このように、ステップＳ１６２０のルックアップ処理は、現在のパレットエントリのレンダリングタグに基づいて、指定のIOT１７０に適する適切なハードウエア及び／又はソフトウエア処理制御を提供するIOT指定メタビット値を戻す。

【０１８７】ステップＳ１６２０でメタビット値が得られると、制御がステップＳ１６３０へ進み、そこで現在のパレットエントリに対するメタビット値が図３１に示されるチャネルデータ構造のメタビットチャネルの次に利用できる位置に格納される。

【０１８８】次に、ステップＳ１６４０において、現在のカラー参照は、それが最後のカラー参照であるか否かを知るためにチェックされる。そうでないならば、制御がステップＳ１６００へ戻り、そこで再び次のカラー参照が現在のカラー参照として選択される。しかしながら、現在のカラー参照が最後のカラー参照である場合、制御がステップＳ１６５０へ進む。

【０１８９】ステップＳ１６５０において、メタビットチャネルに格納されたメタビットデータが圧縮されてチャネルデータ構造のメタビットチャネルに格納される。次に、制御がステップＳ１６６０を介してステップＳ７３０へ戻る。

【０１９０】図２３は、図１６のステップＳ１０７０のプリミティブマスキングオブジェクト処理のより詳細な記述を示す。図２３に示されるように、プリミティブマスキングオブジェクト処理はステップＳ１７００で開始する。ステップＳ１７００において、現在のプリミティブマスキングオブジェクトは、それがビットマッププリミティブであるか否かを決定するためにチェックされる。上述のように、プリミティブオブジェクトは、ボックス、ビットマップ等である。もしそうならば、制御がステップＳ１７１０に進み、そこでこのビットマッププリミティブがマスクビットマップにブリットされ（Blitted）、このビットマッププリミティブがブリットされる位置に予め格納されたデータを上書きする。”ブリッティリング（Blitting）”は、”ビットレベルブロック転送”又は”BITBLT”処理と呼ばれ、バイト又はワード境界ではなくてビット境界上のメモリブロックの変更を可能とする従来の技術である。次に、制御がステップＳ１７１０からステップＳ１７２０へ進む。現在のプリミティブマスキングオブジェクトがビットマッププリミティブではない場合、制御がステップＳ１７００からステップＳ１７２０へ進む。

【０１９１】次に、ステップＳ１７２０において、現在のプリミティブマスキングオブジェクトは、ページ境界ボックスよりもより制限的なクリッピングオブジェクトが有効であるか否かを決定するためにチェックされる。そのようなクリッピング領域がアクティブでない場合、制御がステップＳ１７３０へ進み、そこでこのオブジェクトはこのオブジェクトに対する最初のスキャンライン

に対応するスキャンラインオブジェクトリストへ追加される。即ち、このオブジェクトは、それが最初に表れるスキャンラインのスキャンラインオブジェクトリストへのみ追加される。次に、ステップＳ１７３０から、制御がステップＳ１７８０へ進む。

【０１９２】しかしながら、ページ境界ボックス以外のクリッピング領域が有効である場合、制御がステップＳ１７２０からステップＳ１７４０へ進む。ステップＳ１７４０において、現在のプリミティブマスキングオブジェクトは、それが既存ではあるが不完全なスイープの部分であるか否かを決定するためにチェックされる。もしそうならば、制御がステップＳ１７４０からステップＳ１７５０へ進む。ステップＳ１７５０において、現在のプリミティブマスキングオブジェクトが現在のクリッパーオブジェクトのスイープサブアイテムへ追加される。既存ではあるが、不完全なスイープの一つのタイプは、ボックスのような単純なプリミティブオブジェクトである。現在のプリミティブマスキングオブジェクトが既存のプリミティブオブジェクトに隣接して発見される場合、新たなスイープオブジェクトが生成され、既存で現在のプリミティブオブジェクトは、スイープのs２フィールドへリンクされる。この新たな不完全なスイープオブジェクトはクリッパーのサブアイテムになる。次に、制御がステップＳ１７６０に進み、そこで現在のクリッパーオブジェクトの完全性属性が更新される。現在のクリッパーオブジェクトの完全性属性は、十分なプリミティブマスキングオブジェクトが現在のクリッパーオブジェクトに対応するスイープサブアイテムに追加されて、現在のクリッパーオブジェクトを完全に充填（フィル）したか否かを指示する。ステップＳ１７６０から、制御が再びステップＳ１７８０へ進む。

【０１９３】現在のプリミティブマスキングオブジェクトが既存ではあるか不完全なスイープの部分でないならば、ステップＳ１７４０からステップＳ１７７０に進み、そこで現在のプリミティブマスキングオブジェクトが現在のクリッパーオブジェクトのアイテムリストへ追加される。ステップＳ１７７０から、制御がステップＳ１７８０へ進む。ステップＳ１７８０において、制御がステップＳ１０５０へ戻される。

【０１９４】図２４は、図１９のステップＳ１３２０の次の可視ラン識別処理のより詳細な説明が示される。図２４に示されるように、次の可視ラン識別処理は、ステップＳ１８００で開始する。ステップＳ１８００において、変数”thisrun”は、整列されたランリストに残る次のラン へ初期化される。変数”currentend”が次のランの終りにセットされる。

【０１９５】ステップＳ１８００から、制御がステップＳ１８１０へ進む。ステップＳ１８１０において、整列されたランリストは、それが空であるか否かを知るためにチェックされ、変数”thisrun”は、それが変数”currentend”の後に開始するか否かを知るためにチェックされる。これらの両方が、偽であるならば、制御がステップＳ１８２０に進み、そこで”thisrun”によって参照されるランは、それによって参照されるオブジェクトが変数”highestrun”によって指示されるランセグメントの層の上にある層を有するか否かを知るためにチェックされる。即ち、”highestrun”は、ランセグメント即ちランの部分を指し、それは開始位置、終了位置、前景オブジェクト及び潜在的にチェーン化された背景オブジェクトのリストを有する。”thisrun”によって参照されるオブジェクトの層が、”highestrun”によって参照されるランセグメントの前景層の上でない場合、次に、”thisrun”によって参照されるオブジェクトは”highestrun”によって参照されるランセグメントの前景の直下にある。この場合、制御がステップＳ１８３０へ進み、そこで”thisrun”及びそれに関連するオブジェクトが新たな直下のランとして処理される。ステップＳ１８３０から、制御がステップＳ１８６０へ進む。

【０１９６】しかしながら、”thisrun”によって参照されるオブジェクトの層が”highestrun”の前景層の上であるならば、制御がステップＳ１８４０へ進む。ステップＳ１８４０において、”highestrun”に対する開始位置は、それが”currentstart”によって指示される開始位置の後に始まるか否かを知るためにチェックされる。

【０１９７】これが真でない場合、制御がステップＳ１８５０へ進み、そこでラン”thisrun”が処理されて新たな最も高いランとなる。ステップＳ１８５０から、制御がステップＳ１８６０へ進む。ステップＳ１８６０において、新たな”thisrun”が整列されたランリストから得られる。次に、制御がステップＳ１８６０からステップＳ１８１０へ戻る。しかしながら、ステップＳ１８４０において、”thisrun”が”currentstart”の後で開始するならば、制御がステップＳ１８４０からステップＳ１８７０へ進む。ステップＳ１８７０において、変数”currentend”によって指示された終了位置が”thisrun”の開始位置と等しくなるようにセットされる。次に、制御がステップＳ１８７０からステップＳ１８８０へ進む。

【０１９８】同様に、ステップＳ１８１０において、テストの何れかが真ならば、制御がステップＳ１８８０へ進む。ステップＳ１８８０において、識別されたランが処理される。次に、ステップＳ１８８０から、制御がステップＳ１８９０を介してステップＳ１３３０へ戻る。

【０１９９】平滑ランリスト（Flatten Run List）処理の初めに、現在のスキャンライン上のアクティブなランのリストがある。このリストは、それらの最左点に基づいて、左から右へ整列される。平滑ランリスト処理は、左から右の順に、スキャンライン上の可視の各ランの各部分又はセグメントを正確に述べるコマンドのスト

55

リームを生成する。この処理のきわめて重要な部分は、ページの左端で始まって、全体的に可視のランの最も長い次の部分を識別することである。これは、ステップＳ１３２０の次の可視セグメント識別（Identify Next VisiBle Segment）処理によって達成される。その基本的アプローチは単純である。ランがそれらの開始位置によって整列されるので、それらは、”アクティブランリスト”に追加されることができ、ランのリストは、スキャンラインに沿うこの点でアクティブであると現在考えられている。ランはスキャンラインに沿ってそれらの開始位置によって整列されるので、それらは、それらの開始位置に到達された時にアクティブランリストに追加されることができ、スキャンラインに沿う位置がそれらの終了位置を越えて前進した時に破棄され得る。次に、最も高いターゲット値を有するランが、”頂部（on-top）として識別される。しかしながら、二つのキーファクタ、性能ファクタ及び透明ビットマップオブジェクトは、処理を複雑にする。

【０２００】幾つかのグラフィカル構造は、この単純なタイプの処理を困難にするPDLにおいて際立っている。一例は、放射状スイープとして知られているものである。放射状スイープは、最下層の大きなオブジェクトと、より下の層上の漸進的により小さいオブジェクトとよりなり、あらゆるより上のオブジェクトは、そのようなより下のオブジェクトの全ての内側に完全に或いは実質的に入る。ページ上の各このようなスイープは、数百の層を持つことができる。一つの問題は、オブジェクトのスタックの頂部又はその近くに、アクティブランリスト中に見られる非常に多くのオブジェクトがある。ここで表される例示の実施の形態は、図２４のステップＳ１８３０の直下チェーン化処理を導入する事によって略完全に軽減する。直下チェーン化は、それがより高い層によって覆い隠される時に、現在のより高いランが終了した後、より低い層のランが再出現する限り、それを覆い隠すランへリンクする事によってアクティブランリストから現在の最も高い層を除去する。もしより高いランの前により低いランが終了する場合、より低いランは、それが覆い隠される点で完全に破棄され得る。このように、一時的に覆い隠されたオブジェクトは、アクティブランリストに配される代わりに、チェーン化されるので、アクティブランリストは非常に短く維持される。このように、放射状スイープの全てのランは、現在の頂部ランの直下の長いチェーン内にある。より上のランが終了すると、そのランの直下のチェーンの最初のランがアクティブランリストへ追加される。新たなランは、その点で頂部にない整列されたランリストに遭遇すると、それは、適切な層で直下のチェーンに追加される。即ち、それは、自身よりも下の層を有するランの上方であって且つより高い層を有するランの下方へ追加される。更に、ランは、直下のチェーンから除去され、且つそれらが新たに挿入されたランの直下にあり且つ挿入がなされる前に終了することが知られるやいなや、破棄される。このことは、それらのランが右へのある点でもはや可視にならないことを意味する。

56

【０２０１】第２の複雑さは、透明なビットマップオブジェクトによって引き起こされる。ビットマップオブジェクトは、透明の特質を有し、そこで”１”ビットが着色されるが”０”ビットは透明（クリア）である。即ち、ビットマップオブジェクトの直下のオブジェクトのカラーは、透けて見える。この両者は、直下のチェーン化を複雑にし、マスクチャネルの使用をコンフリクトする。後者の困難さは、図１９の識別ラン処理ステップＳ１３５０及びＳ１３８０の部分として後述される技術で処理される。チェーン化の困難さは、頂部ランが透明である時は常に直下チェーンを空にしてアクティブランリストへ戻し、透明なランの全てがアクティブランリストから除去されると再び直下のチェーンを再構築することによって例示の実施の形態において処理される。

【０２０２】図２９は、メモリ１５０に格納された汎用オブジェクトデータ構造１５５０の汎用化された表現を示す。汎用オブジェクト構造１５５０は、次のリンクフィールド１５５１０、層フィールド１５５２０、パレットエントリポインタフィールド１５５３０、オブジェクトクラス指定手順フィールド１５５４０（上述のフィールドは固定フィールドである）、オブジェクトデータフィールド１５５５０、及びオブジェクトクラスで変化するこれらの数とタイプよりなる。具体的には、次のリンクフィールド１５５１０は、他の一つのオブジェクトの次のリンクフィールド１５５１０を指すことによってオブジェクトのリストを形成するために使用される。このメカニズムによって、図２８に示されるスキャンラインオブジェクトリスト１５１０が形成される。

【０２０３】層フィールド１５５２０は、ページから垂直に延びるＺ軸に沿ってオブジェクトの相対高さをエンコードするために使用される。即ち、ページ画像を述べるPDLファイルは、PDLファイルに先に定義されるページ画像のグラフィカルオブジェクトがファイルに後で定義されるそれらのオブジェクトによって覆い隠されるように構成される。この先に定義されたオブジェクトは、ページ画像を形成するオブジェクトのスタックにＺ軸に沿ってより低いように述べられることが出来る。同様に、PDLファイルに後で表れるオブジェクトは、Ｚ軸に沿ってより高いように述べられることが出来る。層フィールド１５５２０は、スタックのオブジェクトのこの相対的高さをエンコードする。各新たなオブジェクトは、図１６のプリミティブマスキングオブジェクトステップＳ１０７０処理の間、連続的に高くなるレベル値を与えられる。

【０２０４】パレットエントリポインタフィールド１５５３０は、メモリ１５０のパレットデータ構造１５２０

57

中のエントリへのポインタである。参照されたパレットエントリは、一定カラーデータやサンプリングされた画像データへ制限されず、図１５に示されるカラー演算子処理ステップＳ４６０の結果として生成されるパレットエントリであってもよい。オブジェクトクラス指定手順フィールド１５５４０は、一つのオブジェクトクラスから他のオブジェクトクラスへ手順の詳細な動作において変化する手順の収集物へのポインタである。このように、同じタイプ又はクラスを有する全てのオブジェクトのオブジェクトクラス指定手順フィールド１５５４０は、同じ手順を示す。指定オブジェクトクラスの手順は、そのオブジェクトクラス指定のデータへ正確にアクセスできる。

【０２０５】図２９にも示される、ボックスリストクラスデータフィールド１５５６０に対するオブジェクト指定クラスデータフィールドは、リンクされたボックスポインタフィールド１５５６１及び現在のボックスポインタフィールド１５５６２から成る。リンクされたボックスポインタフィールド１５５６１は、オブジェクトを共に形成する一連のボックスを指す。この一連ののボックスの各ボックスは、この一連のボックスの次のボックスへのリンク及びこのボックスの左下及び右上コーナーの位置より成る。従来公知の幾つかの技術は、このような一連のボックスに表され得るボックスへ有用な制約を配する。幾つかの制約は、このような連なるボックスによって述べられるクリッピング領域に作用するクリッピング手順の性能を増加するのに有用である。このように、ボックスリストクラス１５５６２の現在のボックスポインタフィールド１５５６２は、クリッピング及び他の手順に便利なものとして提供される。

【０２０６】ビットマップ指定クラスデータフィールド１５５７０は、ビットマップオブジェクトポインタフィールド１５５７１を含み、このビットマップは、その自身の境界ボックスデータを有する。ビットマップに１の値を有するデータビットは、そのビットで表される点におけるオブジェクトがパレットエントリポインタ１５５３０によって参照されるカラーでプリントされるべきであると言うことを指示する。０の値を有するデータビットは、クリアフィールド１５５７４の値に依存して、二つの交互する意味の一方をとる。クリアフィールド１５５７４が偽を表す０の場合、マップの０のデータビットは白を表す。クリアフィールド１５５７４が真を表す非０である場合、マップの０のデータビットは、ビットマップがそれらの点で透明であり、カラーがビットマップオブジェクト下でオブジェクトによって決定されることを表す。

【０２０７】アウトラインオブジェクトポインタフィールド１５５７２は、ビットマップオブジェクト１５５７０のアウトライン又は境界を表す。参照されるアウトラインオブジェクトは、一般にボックスリストクラスオブジェクトである。このように、複雑な形状が表されることができ、ビットマップ１５７７０自体が矩形であることが可能である。背景オブジェクトポインタフィールド１５５７３は、クリアフィールド１５５７４によって指示されるように、ビットマップが透明である場合、平滑化処理の間使用される。

58

【０２０８】クリップ指定クラスオブジェクトデータフィールド１５５８０は、クリッパーオブジェクトポインタ１５５８１、オブジェクトインサイドクリッパー領域ポインタ１５５８２及び完全性属性１５５８３を含む。クリッパーオブジェクトポインタ１５５８１は、図１７の”setclip”演算子ステップＳ１１７０によってセットされるように、クリッパー領域の形状を指定するためにボックスリストクラスオブジェクトを指す。オブジェクトインサイドポインタ１５５８２は、このクリップクラスオブジェクトによって表されるクリッパー領域が現在のクリッピング領域である時、図２３に示されるように、処理プリミティブマスキングオブジェクトステップＳ１７７０で処理される。完全性属性フィールド１５５８３は、クリッピング領域を表す現在のクリップクラスオブジェクトがクリッピング領域の境界ボックスを充填（フィル）しながら部分的に完全なスイープがいかに完全にどのような方法で収集されるべきかをエンコードするために使用される。

【０２０９】スイープ指定オブジェクトクラスデータフィールド１５５９０は、s1オブジェクトポインタフィールド１５５９１、s2オブジェクトポインタフィールド１５５９２、スイープ変化レートフィールド１５５９３、及び着色ランプロジューサ方法フィールド１５５９４より成る。ｓ１オブジェクトポインタフィールド１５５９１は、スイープのアウトラインオブジェクトを指すために主に使用され、ボックスリストクラスオブジェクトは、スイープからなるオブジェクトが収集されていた時に有効であるクリッピング領域を表す。ｓ２オブジェクトポインタフィールド１５５９２は、クラススイープの第２のオブジェクトを指し、そのｓ１及びｓ２オブジェクトは、スイープの変化する着色されたスライスを表す単純なオブジェクトのリンクされたリストの二つの端を指す。スイープの変化レートフィールド１５５９３は、図１７に示されるように、グラフィカル状態演算子処理ステップＳ１１５０に示されるように、クリッピング領域のインサイドでオブジェクトを収集するために使用されるクリップクラスオブジェクトがスイープクラスオブジェクトへ変換される時に計算される。それは、上述の動作の”スローサンプリングチャネル”モードの使用を保証するのに十分な頻度でスイープが変化するカラーであるか否かを決定するために使用される。

【０２１０】着色（カラー）ランプロジューサ方法１５５９４は、収集されたスイープのタイプ固有の手順である。垂直に変化するスイープ、水平に変化するスイー

プ、及び種々の他の特徴を有するスイープは、異なるカラーランプロジューサ方法手順である。この手順は、図２０に示される生成スイープコマンド及びカラーステップＳ１４３０の間に呼び出され、このフィールド１５５９４が指すことが出来る各着色スライス毎に、コマンド及びカラーを生成する。

【０２１１】図２５は、図２０のステップＳ１４２０のノーマルコマンド及びカラー生成処理のより詳細な説明を示す。ノーマルコマンド及びカラー生成処理は、ステップＳ１９００で開始し、そこでオブジェクトのカラークラスは、そのオブジェクトがそのカラーとしてサンプリングされた画像を有するか否かを決定するためにチェックされる。もし否ならば、制御がステップＳ１９１０へ進み、このステップでカラーレジスタの一つにこのランに対する適切なカラーデータ及びメタビット値がロードされることを確保する。このランの前景に対するパレット参照がカラー／メタビットレジスタ０－２に対するシャドーレジスタの各々に保持されているパレット参照と比較されて、オブジェクトによって参照されるカラーがカラーレジスタの一つに以前にロードされていたか否かを決定する。これがそうでない場合、カラー／メタビットレジスタ０－２の一つが選択されて以下のステップで発生されるコマンドによってカラーデータがロードされる。この選択は、”最低使用頻度（least recently used）”処理や他の同様又は等価な処理のようなあらゆる従来の処理を使用できる。最後に、このランに対応するパレットに対する参照は、一定カラーチャネルデータ構造の次に利用できる位置へ出力される。

【０２１２】次に、制御がステップＳ１９２０へ進み、そこでカラーレジスタ選択値A、ロードカラー及びこのカラーレジスタ選択値Aによって選択された特定のカラーレジスタに対する適切なビット値を有するノーマルコマンドが、ステップＳ１９１０で決定された値に基づいて出される。更に、多くの”繰り返し”コマンドは、必要に応じて、ノーマルコマンドにおける６４画素長制限を越えてランを拡張するために発生される。ステップＳ１９２０から、制御がステップＳ１９６０へ進み、そこで制御がステップＳ１４６０へ戻される。

【０２１３】もし、ステップＳ１９００において、オブジェクトのカラーがサンプリングされた画像である場合、制御がステップＳ１９３０へ進み、それによりサンプル画像に対応するメタビット値は、画像画素が表示される時にメタビットレジスタ３へロードされるとを確実にする。シャドーレジスタは、全体のコマンド及びカラーパレット参照発生処理の間メモリ１５０のRAM部分１５１に保持され、それによってカラーレジスタの現在の内容が決定できる。この場合、カラーレジスタ３の値は重要ではない。メタビットレジスタ３の値のみが使用され、カラー画素データがサンプルチャネルによって供給される。シャドーレジスタの内容に基づいて、メタビットレジスタのプレ値がロードされる必要がある場合、カラーパレット参照が一定カラーチャネルの次に利用可能位置へ挿入されて正確なレンダリングタグ及びあまり重要でないカラー値を有するパレットエントリを参照する。

【０２１４】メタビットレジスタ３の値がステップＳ１９３０で検証された後、制御がステップＳ１９４０へ進み、そこでコマンドがサンプルチャネルデータを表示し、適切なサンプル除数をセットアップし、ステップＳ１９３０で決定されたメタビットレジスタをチャネルメモリ部分の次に利用できる位置へロードする。ステップＳ１９２０におけるように、”繰り返し”コマンドが、必要に応じて、ノーマルコマンドの６４画素制限を越えてラン長を拡張するように発生される。

【０２１５】次に、ステップＳ１９４０において、制御がステップＳ１９５０へ進み、そこで丁度発生されたコマンドに対して供給されるサンプル画像データのアドレス及び長さがサンプルポインタチャネルデータ構造の次の利用可能位置へロードされる。DMAコントローラによってアドレス及び長さを使用する事に加えて、図１４のステップＳ７４０の抽出及び格納サンプル画像ポインタ処理、及び抽出、圧縮及び格納サンプリング画像データ処理が、サンプル画像の必要な部分を発見し抽出するためにそのアドレス及び長さを使用する。

【０２１６】次に、ステップＳ１９５０から、制御がステップＳ１９６０を介してステップＳ１４６０へ戻る。

【０２１７】図２６は、図２０のステップＳ１４３０のスイープコマンド及びカラー生成処理のより詳細な記述を示す。スイープコマンド及びカラー命令生成処理は、ステップＳ２０００で始まり、そこでスイープの変化レートは、それがカラーステップ当り２画素以下であるか否かを知るためにチェックされる。実際のコマンド及びカラーを生成するために、オブジェクトデータ構造のスイープ指定データで見つけられるスイープカラーランプロジューサ（生成）方法が呼び出される。この方法は、スイープのタイプによって異なり、スイープの背景スライスを抽出してパレットエントリ及びスイープオブジェクト内の各サブラン毎に開始／終了位置を提供する。例えば、カラーが一つのスキャンラインから次のスキャンラインへのみ変化するスイープは、唯各スキャンラインに対して適切な一つのスライスを見つけ、且つそれをコマンド発生プロセッサに唯１度だけ通知するカラーランプロジューサ方法を有する。次に、コマンド発生プロセッサは、全体のラン長に対するコマンド及びカラー参照を発生する。対照的に、スキャンラインに沿ってカラーからカラーへステップするスイープは、スキャンラインに沿ってオブジェクトの各スライス毎に一つのコマンド及び一つのカラー参照を持たなければならない。スイープの変化レートは、一旦そのスイープが一つのクリッパーオブジェクトから変換されると、クリッパー検証処理の間に計算される。このフィールドは、スイープにおけるカラーステップ毎に使用される画素の平均数を指示する。変化レートがカラーステップ当り２画素以下である場合、制御がステップＳ２０２０へ進み、カラーレジスタ選択B値を使用してカラー／メタビットレジスタ３を選択する一つの単一コマンドを発生する。上述のように、カラーレジスタ選択B値を有するカラー／メタビットレジスタ３を使用する時、一つのカラーがクロックサイクル毎に一定カラーチャネルから読み込まれる。このスローサンプルチャネルを使用する事によって、単一のコマンドのみが、各カラースライス毎に唯一つのカラーパレット参照と共に、要求される。IOTコントローラをセットアップするコマンドビット及びカラーパレット参照は、コマンドチャネルメモリ及び一定カラーチャネルメモリの次に利用可能な位置へ出力される。

【０２１８】次に、制御がステップＳ２０２０からステップＳ２０３０を介してステップＳ１４６０へ進む。

【０２１９】しかしながら、スイープの変化レートが２画素を越えると、制御がステップＳ２０１０へ進む。ステップＳ２０１０において、コマンド及びカラーパレット参照が発生されてコマンドチャネルメモリ及び一定カラーチャネルメモリへロードされなければならない。ステップＳ２０２０の場合のように、制御がステップＳ２０１０からステップＳ２０３０を介してステップＳ１４６０へ進む。

【０２２０】図２７は、ステップＳ１４５０のマスク処理を使用するコマンド及びカラー生成のより詳細な記述を示す。この処理は、コンバイナがノーマルにラン（実行）することを可能とする条件を確立するために、長さ１のコマンドが発行されなければならない特別な場合があるか否かを決定する。このような条件は、透明スイープの両方のカラーをロードすることの必要性、又はマスクを使用して画像の前景と一定カラー前景の間でのスイッチをする時に、メタビット値及び一定カラーの両方をロードすることの必要性を含む。これらの場合の幾つかにおいて、マスクデータ自体の最初（第１）のビットカラーレジスタがロードされる順序を決定するために調べられなければならず、それによってマスクデータの最初の画素によって選択されるカラーがその画素を出すコマンドによってロードされたカラーレジスタを選択する。第２のコマンドを出して、他の必要なカラー／メタビットレジスタをその第２の画素クロックへロードすることが出来るように、その最初のコマンドは１画素の長さを有する。幾つかの場合、マスクデータは、反転されて処理されることが必要である。それは、例えば、マスクを使用して画像の上に一定カラーを選択するために発生する。サンプルチャネルがカラーレジスタ選択A値によって選択されるに過ぎないので、SrcBとコマンド中のSrcBフィールドの間のマスク選択は、この場合において本例のカラーレジスタ選択B値によって選択されなければならない一定カラーを前景に配するために指定されなければならない。

【０２２１】その処理は、ステップＳ２１００において開始し、そこで前景及び背景カラーは、それらの何れがサンプリングされた画像カラーであるかを知るためにチェックされる。肯定の場合、制御がステップＳ２１１０へ進み、そこで画像メタビット及び一定カラーがそのマスクのために検証される。即ち、カラーレジスタ及びメタビットのロード順序は、その処理が正確に動作することを確実にするために決定される。この決定は、カラーレジスタの何れかに必要とされる一定カラーがあるかないかを決定し、且つメタビットレジスタ３のメタビット値が明示的にロードされる必要があるか否かを決定することによってマスクデータの初期ビットを問合せることによってなされる。次に、ハードウエアを適切な状態に適切に初期化するためにここで必要とされるように決定される１画素コマンドの数は、引き続くステップでコマンドを発生するために使用される。

【０２２２】次に、制御がステップＳ２１１０からステップＳ２１２０へ進み、ステップＳ２１１０で決定されたデータを実際に発生する。

【０２２３】次に、制御がステップＳ２１２０からステップＳ２１９０を介してステップＳ１４６０へ戻る。

【０２２４】しかしながら、若し、ステップＳ２１００において、前景及び背景がサンプリングされた画像カラーでないならば、制御がステップＳ２１３０へ進み、そこでシャドーレジスタは、前景及び背景カラーの両方がロードされる必要があるか否かを決定するためにチェックされる。若しそうならば、制御がステップＳ２１３０からステップＳ２１４０へ進み、ロードされる必要のある二つのカラーの内どちらが最初にロードされるべきかを決定する。また、上述のように、これは、マスクデータの最初のビットを調べることによってなされる。最初にロードされる必要があるカラーがステップＳ２１４０で決定されると、制御がステップＳ２１５０へ進み、そこで１画素長コマンドが発生されて、最初のカラーをロードするためにコマンドチャネルメモリの最初に利用できる位置へロードされる。勿論、最初のカラーは、一定カラーチャネルの次に利用できる位置へロードされる。

【０２２５】次に、制御がステップＳ２１７０へ進み、そこで２番目にロードされると決定されたカラーがその二つカラーの間で適切に選択するマスクを有するラン長の残りに対するコマンドを発生する事によってロードされる。このように、第２番目のコマンドは、コマンドチャネルメモリの次に利用できる位置へロードされ、且つ第２番目のパレット参照が一定カラーチャネルの次に利用できる位置へロードされる。

【０２２６】しかしながら、ステップＳ２１３０で前景及び背景カラーの両方がロードされる必要がない場合、制御がステップＳ２１６０へ進み、そこでシャドーレジ

スタは、前景及び背景カラーの一方がロードされる必要があるか否かを決定するためにチェックされる。若しそうならば、制御がステップＳ２１７０へ進む。若しそうでないならば、制御がステップＳ２１８０へ進み、そこでコマンドが発生され、コマンドチャネルメモリの次に利用できる位置へロードされる。しかしながら、両方のカラーがすでにIOTコントローラのレジスタにロードされているので、パレットへの追加の参照が発生されて一定カラーチャネルメモリへロードされる必要がない。次に、ステップＳ２１８０から、制御がステップＳ２１９０を介してステップＳ１４６０へ進む。

【０２２７】図３１は、メモリ１５０のRAM部分１５１の象徴的な図を示す。RAM部分１５１は、カラーパレット１５３０、マスクビットマップ１５６０、パレット１５２０、スキャンラインポインタスタック１５１０、コマンドチャネルメモリ１５３、一定カラーチャネルメモリ１５４、サンプルカラーチャネル１５５、マスクチャネル１５６、及びメタビットチャネル１５７を含む。

【０２２８】図３２は、一つのシステムに統合された、IOTコントローラ１６０、分解システム１３０、及びコマンド命令とデータ発生システムを有し、開始から終了まで順序付けられたデータフロー及び処理手順を有する全体オブジェクト最適化システムの象徴的な図を示す。IOTコントローラ１６０、IOT１７０及び／又は分解システム１３０に利用可能なオブジェクト最適化リソースもまた示される。

【０２２９】上に概説されたように、本発明のユーティリティが本発明のシステム及び方法を含む特定の例示的実施の形態と共に説明されたが、多くの代替え、変更及びバリエーションが当業者には明瞭であることは明白である。従って、上述のように、本発明のシステム及び方法と共に使用される構造は、例示であって本発明を限定する事を意図しない。種々の変更が本発明の精神及び範囲内でなされ得る。

【０２３０】例えば、画像は、その画像を画素のサイズよりも大きなセグメント、即ち領域、にセグメント化する事によって処理され得る。画像のあらゆる特定の領域を露光するために使用されるレーザパワーに対するセットポイントは、所与の領域構成、即ち、その領域内にハーフトーンオブジェクト又はその部分が幾つあるか、及びテキストオブジェクトやラインアートオブジェクト又はそのようなオブジェクトが幾つあるか、に基づくことができる。従って、異なるレーザパワーセットポイントは、異なるタイプのオブジェクト、即ちハーフトーンオブジェクト及びテキスト及び／又はラインアート、の比率に基づいて領域を露光するために使用され得る。例えば、低いセットポイントは、ハーフトーンオブジェクトのみを含む領域と共に使用されることができ、中間のセットポイントは、ハーフトーンオブジェクト及びテキスト及び／又はラインアートオブジェクトを含む領域と共に使用されることができ、高いレーザパワーセットポイントは、テキスト及び／又はラインアートのみを含む領域と共に使用されることができる。

【０２３１】これらの領域は、互いから区別されることができ、且つ公知又は将来開発されるセグメントアルゴリズム、各領域の空間座標に基づいて、セグメントビットを使用する各領域のタイプによって、又は他の公知の又は将来開発される方法によって識別され得る。例えば、プリントシステムユーザは、これらの領域を形成する時に使用されるレーザパワーセットポイントに基づいて、種々の動作モードを使用してプリントされるべき領域を描くためのインターフェースを介して端末でデータを入力できる。

【０２３２】ハーフトーン領域とテキスト及び／又はラインアートとの間の差に基づいてレーザ変調を実行する構造及びデバイスは、例えば、限定されるわけではないが、プリンタ、ファクシミリマシン及び例えばレザーラスタ出力スキャナやＬＥＤ画像バースキャナや発光デバイスのパワーの出力強度を変調するあらゆる公知又は将来開発されるデバイスを使用するデジタル複写機に含まれる。

【０２３３】本発明は、伝統的な方法とは異なるハーフトーン画像をレンダリングする方法を使用する超高精度プリンティングと共に実施できる。超高精度プリンティングは、異なる形状を使用してハーフトーンドットをレンダリングすることによって実行される。カリー（Curry）に付与された米国特許第５４８５２８９が超高精度プリンティングの詳細に開示している。”閾値アレイ”を使用する伝統的なハーフトーン処理方法もまた使用され得る。

【０２３４】最後に、本発明のユーティリティはレーザを使用してプリント画像を生成する種々の例示構造を使用して上述されたが、またそれと同様に、本発明によるシステム及び方法は、インクジェットプリンタと共に使用され得る。従って、露光スッポトのサイズを変化させるためにレーザ変調を使用するのではなくて、ハーフトーン領域を生成する時にインク量制御がテキスト及び／又はラインアートを生成する時に使用されるインク量に比較して異なるセットポイントを必然的に使用する。

【図面の簡単な説明】

【図１】オブジェクト最適化電子サブシステムのブロック図を示す。

【図２】IOTコントローラの第１の実施の形態のブロック図を示す。

【図３】マルチチャネルコンバナの第１の実施の形態を示す。

【図４】FIFO制御及びコマンド／マスクチャネルプロセッサの第１の実施の形態を示す。

【図５】IOTコントローラのバスインターフェースの第１の好ましい実施の形態を示す。

【図６】IOTコントローラの第２の実施の形態を示す。

【図７】IOTコントローラの第３の実施の形態を示す。

【図８】IOTコントローラの第４の実施の形態を示す。

【図９】オブジェクト最適化処理方法の全体のフロー図を示す。

【図１０】オブジェクト最適化レンダリング及び圧縮を使用してプリントデータを準備するためのフロー図を示す。

【図１１】オブジェクト最適化圧縮解除及びレンダリングを使用して結合及びプリントするためのフロー図を示す。

【図１２】オブジェクト最適化レンダリングタグを有するオブジェクトリストを構成するためのフロー図を示す。

【図１３】スキャンラインデータを発生するためのフロー図を示す。

【図１４】リアルタイムデータをローディング及び圧縮するためのフロー図を示す。

【図１５】カラー演算子を処理するためのフロー図を示す。

【図１６】マスキング演算子を処理するためのフロー図を示す。

【図１７】グラフィカル状態演算子を処理するためのフロー図を示す。

【図１８】スキャンラインランリストを発生するためのフロー図を示す。

【図１９】ランリストを平坦化するためのフロー図を示す。

【図２０】現在のランのためにコマンド及びカラーを生成するためのフロー図を示す。

【図２１】一定のカラーデータを抽出、圧縮及び格納するためのフロー図を示す。

【図２２】メタデータを抽出、圧縮及び格納するためのフロー図を示す。

【図２３】基本マスキングオブジェクトを処理するためのフロー図を示す。

【図２４】次の可視ランを識別するためのフロー図を示す。

【図２５】通常のコマンド及びカラーを生成するためのフロー図を示す。

【図２６】掃引コマンド及びカラーを生成するためのフロー図を示す。

【図２７】マスクデータを使用してコマンド及びカラーを生成するためのフロー図を示す。

【図２８】メモリ１５０に格納されたデータ構造の図を示す。

【図２９】スキャンライン上の各オブジェクト毎の汎用構造を示す。

【図３０】カラーパレットに対する汎用構造を示す。

【図３１】汎用チャネルデータ構造を示す。

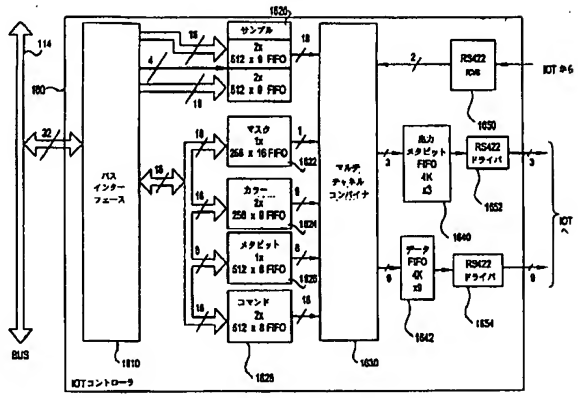
【図３２】システムフロー図及びリソース図を示す。

【図３３】システムフロー図及びリソース図を示す。

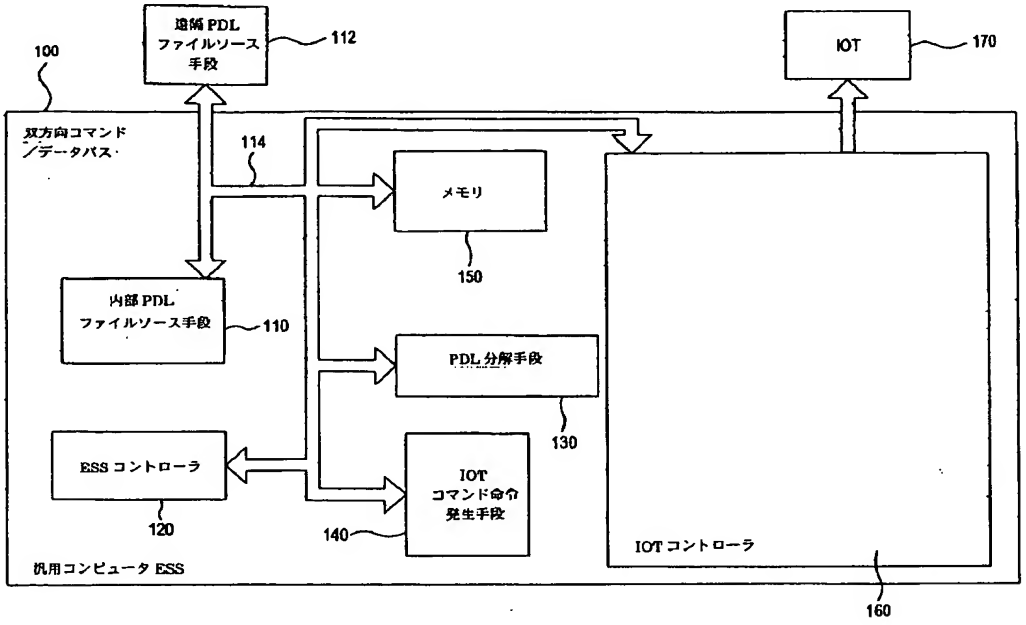
【図３４】IOT内の画像処理システムの第１の好ましい実施の形態のブロック図を示す。

【図３５】オブジェクト最適化プリント測定及び調節システムの第１の実施の形態のブロック図を示す。

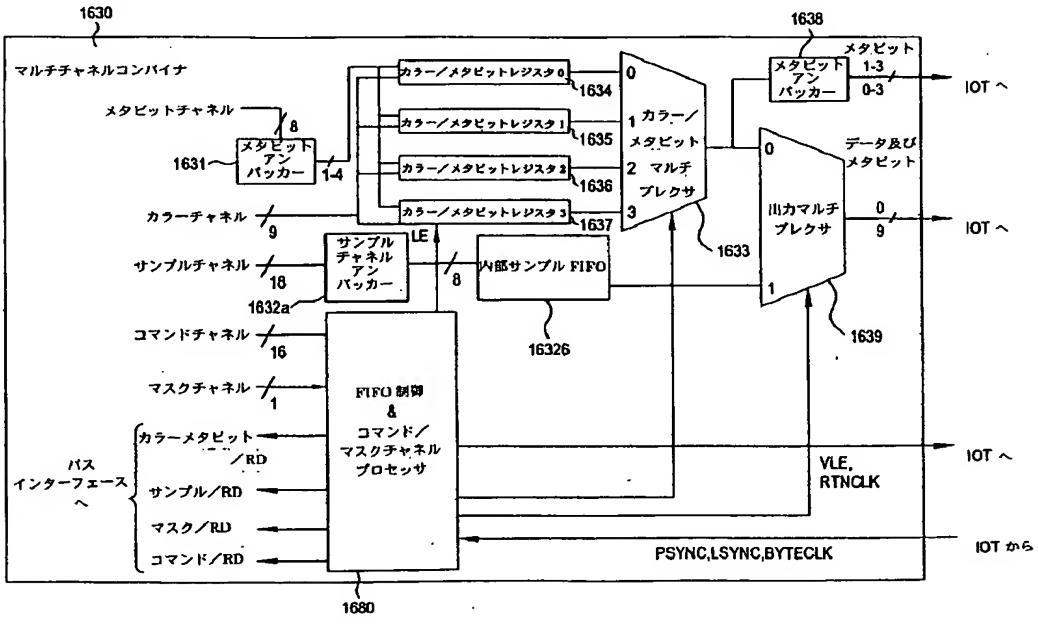
【図２】



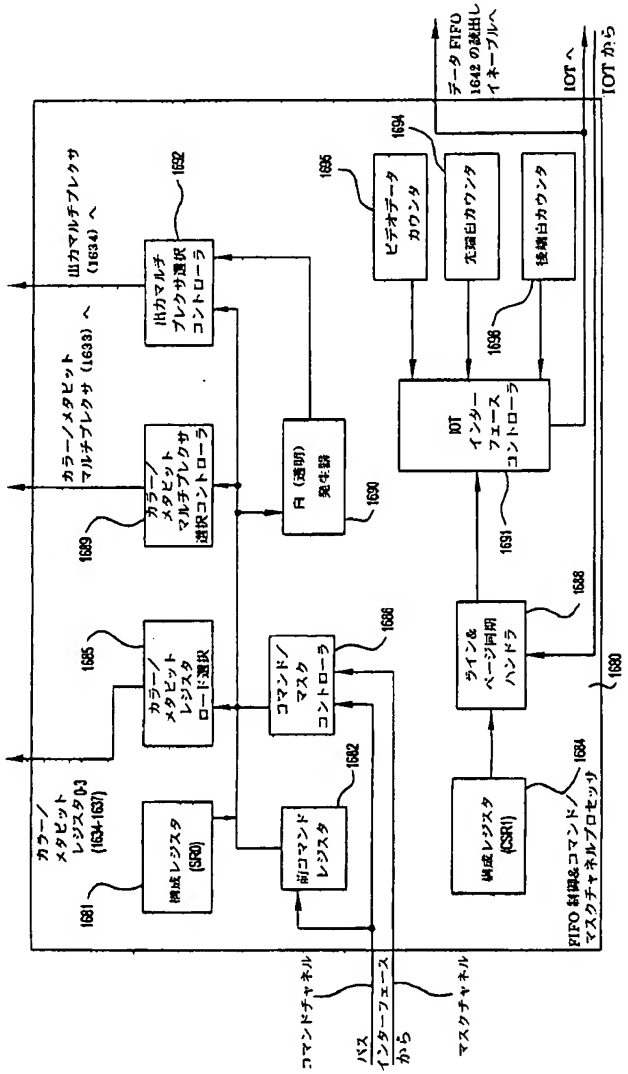
【図1】


遠隔 PDL
ファイルソース
手段
112
100
IOT
170
双方向コマンド
／データバス
114
メモリ
150
内部 PDL
ファイルソース手段
110
PDL 分解手段
130
ESS コントローラ
120
IOT
コマンド命令
発生手段
140
IOT コントローラ
160
汎用コンピュータ ESS

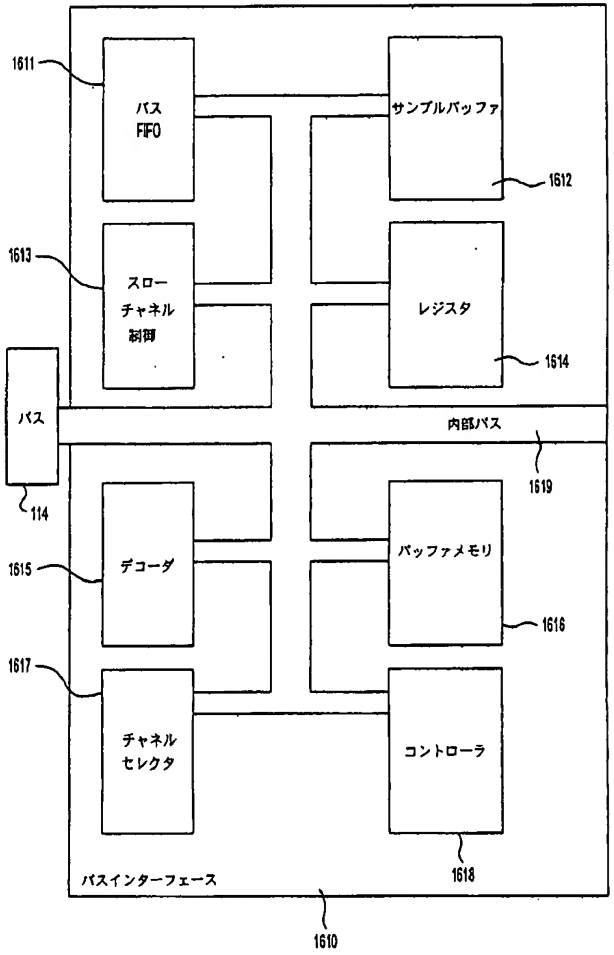
【図3】


1630
マルチチャネルコンバイナ
1638
メタビット
アン
パッカー
1-3
0-3
IOT へ
カラー／メタビットレジスタ 0
1634
メタビットチャネル
8
メタビット
アン
パッカー
1631
1-4
カラー／メタビットレジスタ 1
1635
カラー／メタビットレジスタ 2
1636
カラー／
メタビット
マルチ
プレクサ
1633
データ及び
メタビット
出力マルチ
プレクサ
0
9
IOT へ
カラーチャネル
9
カラー／メタビットレジスタ 3
1637
LE
サンプル
チャネル
アン
パッカー
サンプルチャネル
18
1632a
8
内部サンプル FIFO
16326
1639
コマンドチャネル
16
マスクチャネル
1
FIFO 制御
&
コマンド／
マスクチャネル
プロセッサ
カラーメタビット
／RD
サンプル／RD
マスク／RD
コマンド／RD
バス
インターフェース
へ
VLE,
RTNCLK
IOT へ
PSYNC,LSYNC,BYTECLK
IOT から
1680

【図4】


データFIFO 1642の読出しイネーブルへ
IOTへ
IOTから
ビデオデータカウンタ
1694
1695
先端出カウンタ
後端出カウンタ
1698
IOTインターフェースコントローラ
1691
1632
1カマルチプレクサ選択コントローラ
1カマルチプレクサ(1634)へ
カラー/メタビットマルチプレクサ(1638)へ
カラー/メタビットマルチプレクサ選択コントローラ
1689
FI(透明)発生器
1690
1685
カラー/メタビットレジスタロード選択
1686
コマンド/マスクコントローラ
1688
ライン&ページ同期ハンドラ
1680
カラー/メタビットレジスタ(0:3)(1634-1637)
構成レジスタ(SR0)
1681
前コマンドレジスタ
1682
構成レジスタ(CSR1)
1684
FIFO制御&コマンド/マスクチャネルプロセッサ
コマンドチャネル
バスインターフェースから
マスクチャネル

【図5】


1611
バスFIFO
サンプルバッファ
1612
1613
スローチャネル制御
レジスタ
1614
バス
114
内部バス
1619
デコーダ
バッファメモリ
1615
1616
1617
チャネルセレクタ
コントローラ
1618
バスインターフェース
1610

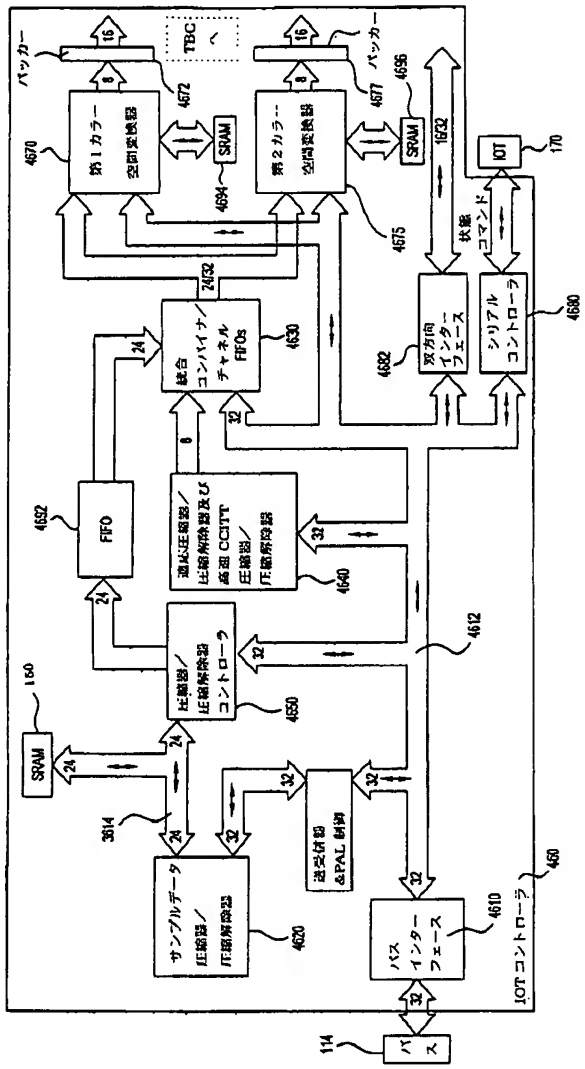
【図6】

IOTコントローラ
260
SRAM
160
2614
2630
170
116
2640
2650
2620
2610
2612
2670
2672
SRAM

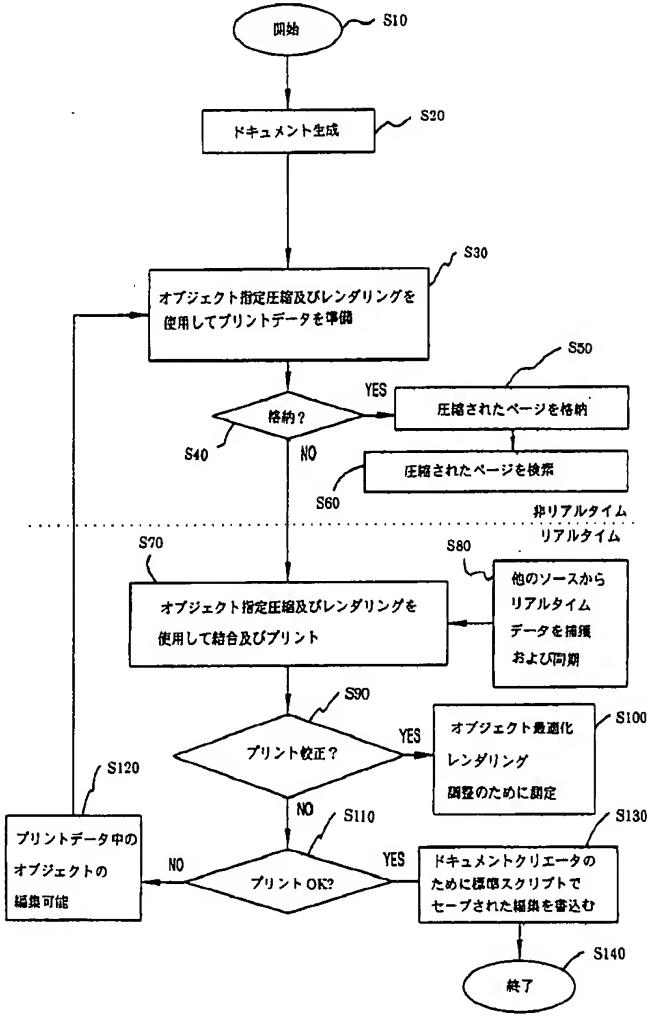
【図7】

IOTコントローラ
360
SRAM
150
3614
3630
3640
3610
3612
3680
3660
3670
170
IOT

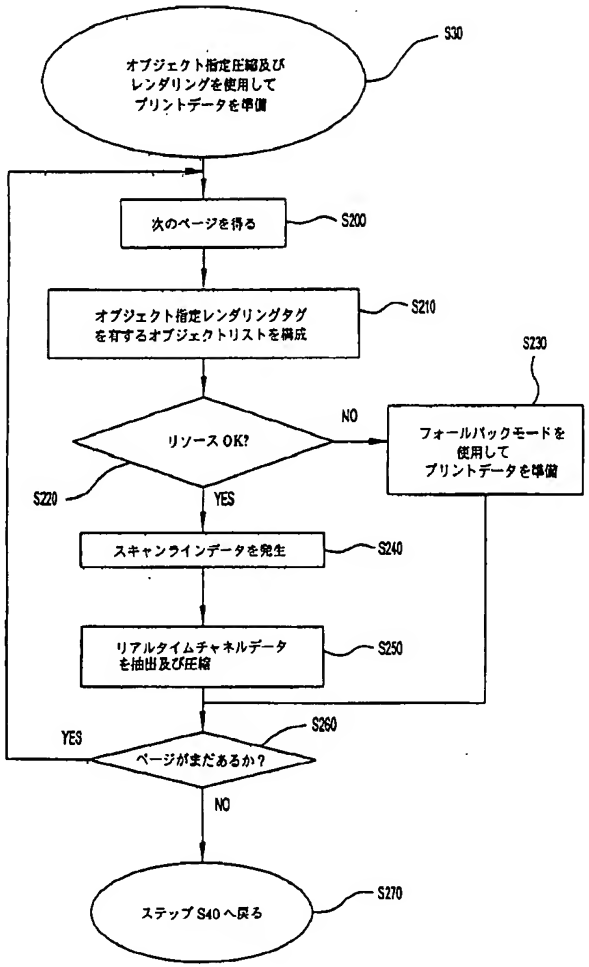
【図８】


パッカー
TBC
パッカー
第１カラー 空間変換器
第２カラー 空間変換器
SRAM
SRAM
IOT
状態
コマンド
シリアル コントローラ
双方向 インター フェース
結合 コンバイナ／ チャネル FIFOs
FIFO
圧縮器／ 圧縮解除器
コントローラ
SRAM
サンプルデータ 圧縮器／ 圧縮解除器
バス インター フェース
バス
IOTコントローラ

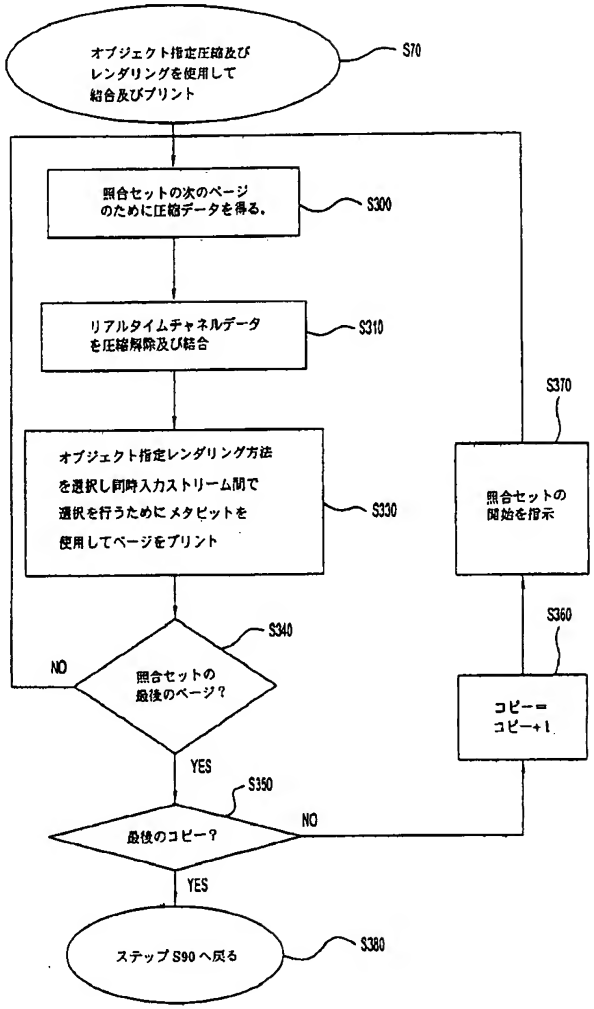
【図９】


開始
S10
ドキュメント生成
S20
オブジェクト指定圧縮及びレンダリングを使用してプリントデータを準備
S30
格納？
YES
S40
圧縮されたページを格納
S50
NO
圧縮されたページを検索
S60
非リアルタイム
リアルタイム
S70
オブジェクト指定圧縮及びレンダリングを使用して結合及びプリント
S80
他のソースからリアルタイムデータを捕獲および同期
S90
プリント較正？
YES
オブジェクト最適化レンダリング調整のために固定
S100
NO
S110
プリントOK?
NO
S120
プリントデータ中のオブジェクトの編集可能
YES
ドキュメントクリエータのために標準スクリプトでセーブされた編集を書込む
S130
終了
S140

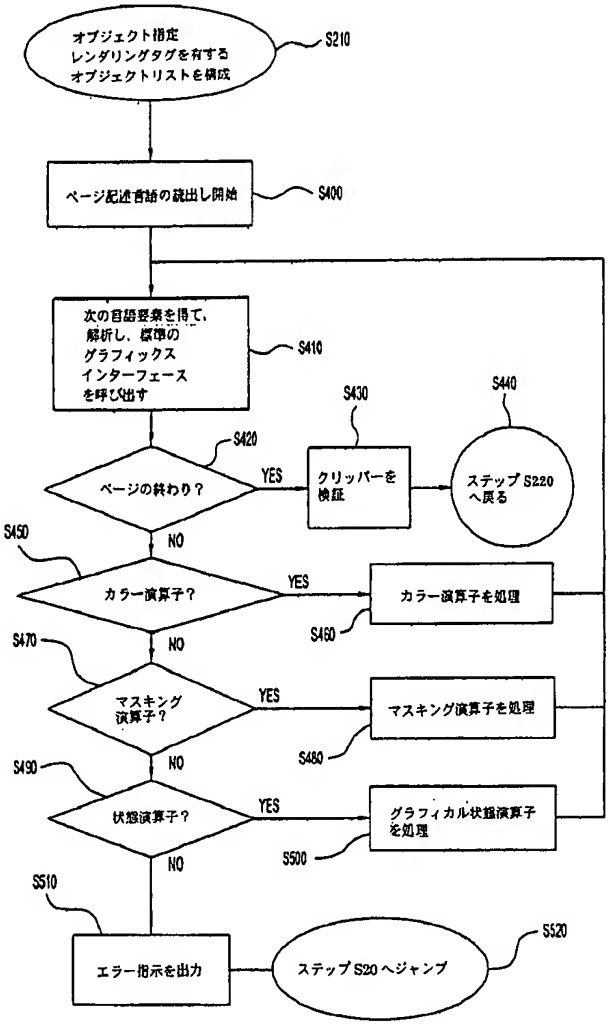
【図１０】


オブジェクト指定圧縮及び
レンダリングを使用して
プリントデータを準備
S30
次のページを得る
S200
オブジェクト指定レンダリングタグ
を有するオブジェクトリストを構成
S210
リソース OK?
NO
S220
YES
フォールバックモードを
使用して
プリントデータを準備
S230
スキャンラインデータを発生
S240
リアルタイムチャネルデータ
を抽出及び圧縮
S250
ページがまだあるか？
S260
YES
NO
ステップ S40 へ戻る
S270

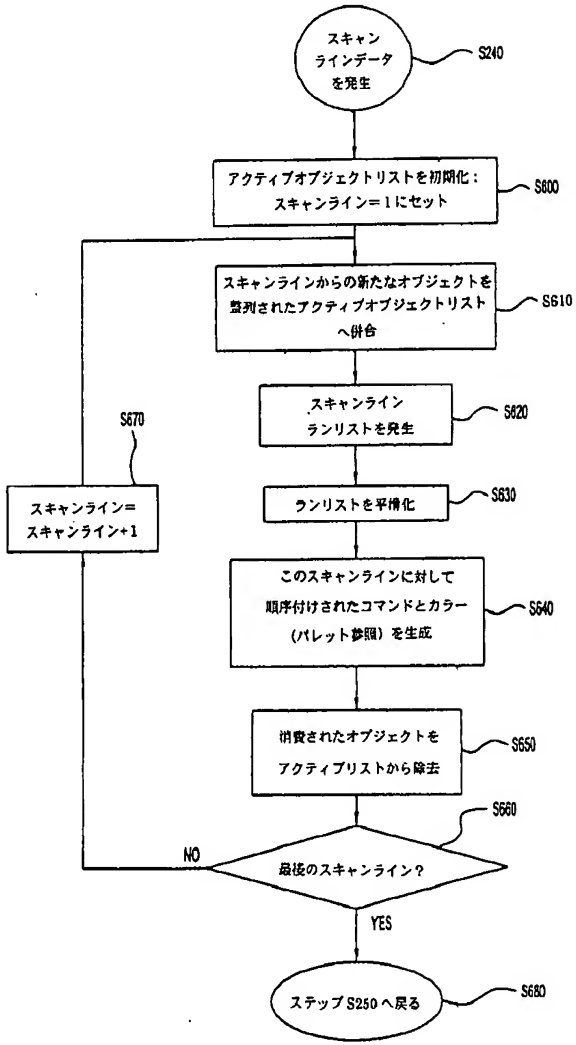
【図１１】


オブジェクト指定圧縮及び
レンダリングを使用して
結合及びプリント
S70
照合セットの次のページ
のために圧縮データを得る。
S300
リアルタイムチャネルデータ
を圧縮解除及び結合
S310
オブジェクト指定レンダリング方法
を選択し同時入力ストリーム間で
選択を行うためにメタビットを
使用してページをプリント
S330
照合セットの
最後のページ？
S340
NO
YES
最後のコピー？
S350
NO
YES
ステップ S90 へ戻る
S380
照合セットの
開始を指示
S370
コピー＝
コピー＋１
S360

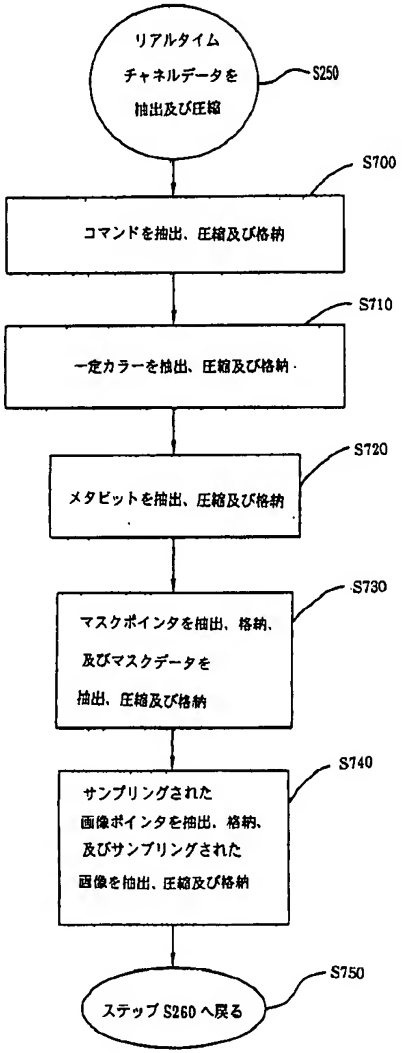
【図12】


オブジェクト指定
レンダリングタグを有する
オブジェクトリストを構成
S210
ページ記述言語の読出し開始
S400
次の目語要素を得て、
解析し、標準の
グラフィックス
インターフェース
を呼び出す
S410
S420
ページの終わり？
YES
S430
クリッパーを
検証
S440
ステップS220
へ戻る
NO
S450
カラー演算子？
YES
カラー演算子を処理
S460
NO
S470
マスキング
演算子？
YES
マスキング演算子を処理
S480
NO
S490
状態演算子？
YES
グラフィカル状態演算子
を処理
S500
NO
S510
エラー指示を出力
ステップS20へジャンプ
S520

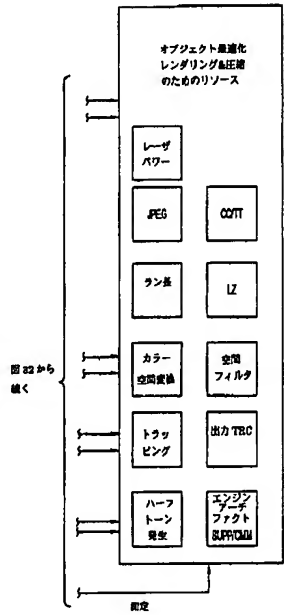
【図13】


スキャン
ラインデータ
を発生
S240
アクティブオブジェクトリストを初期化：
スキャンライン＝1にセット
S600
スキャンラインからの新たなオブジェクトを
整列されたアクティブオブジェクトリスト
へ併合
S610
スキャンライン
ランリストを発生
S620
ランリストを平滑化
S630
このスキャンラインに対して
順序付けされたコマンドとカラー
（パレット参照）を生成
S640
消費されたオブジェクトを
アクティブリストから除去
S650
S660
最後のスキャンライン？
NO
YES
S670
スキャンライン＝
スキャンライン+1
ステップS250へ戻る
S680

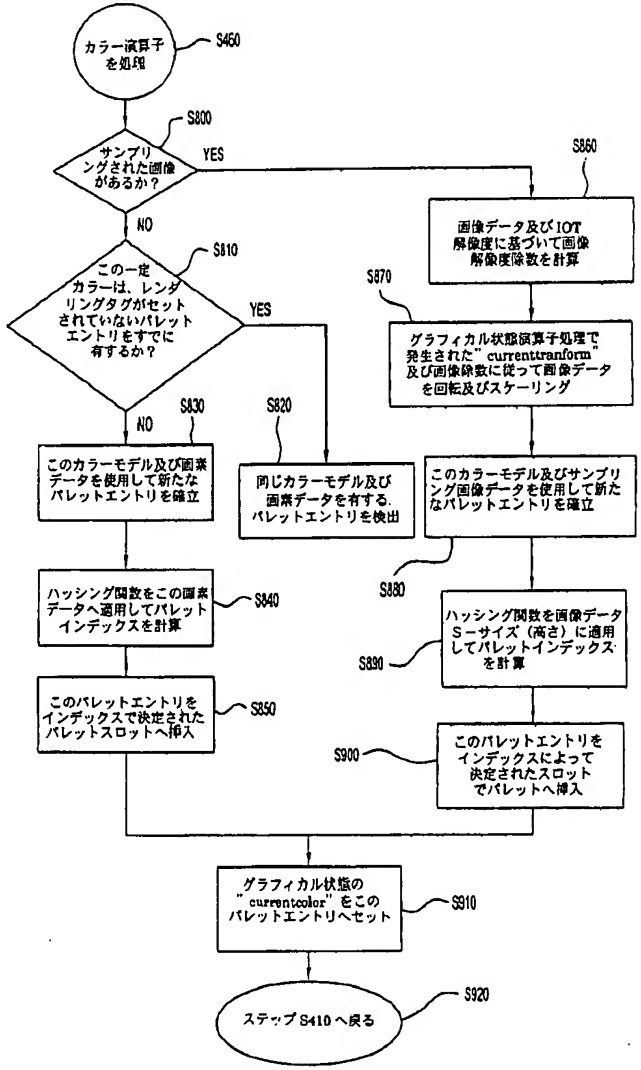
【図14】


リアルタイム
チャネルデータを
抽出及び圧縮
S250
S700
コマンドを抽出、圧縮及び格納
S710
一定カラーを抽出、圧縮及び格納
S720
メタビットを抽出、圧縮及び格納
S730
マスクポインタを抽出、格納、
及びマスクデータを
抽出、圧縮及び格納
S740
サンプリングされた
画像ポインタを抽出、格納、
及びサンプリングされた
画像を抽出、圧縮及び格納
S750
ステップS260へ戻る

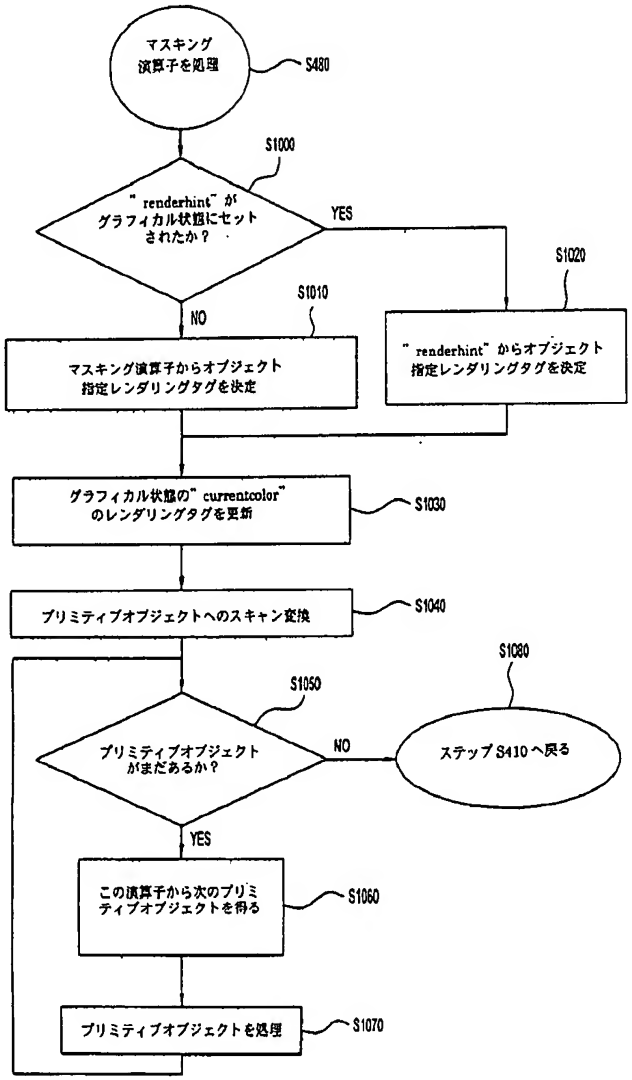
【図33】


オブジェクト最適化
レンダリング&圧縮
のためのリソース
レーザ
パワー
JPEG
CCITT
ラン長
LZ
カラー
空間変換
空間
フィルタ
トラッ
ピング
出力TRC
ハーフ
トーン
発生
エンジン
アーチ
ファクト
図32から
続く
固定

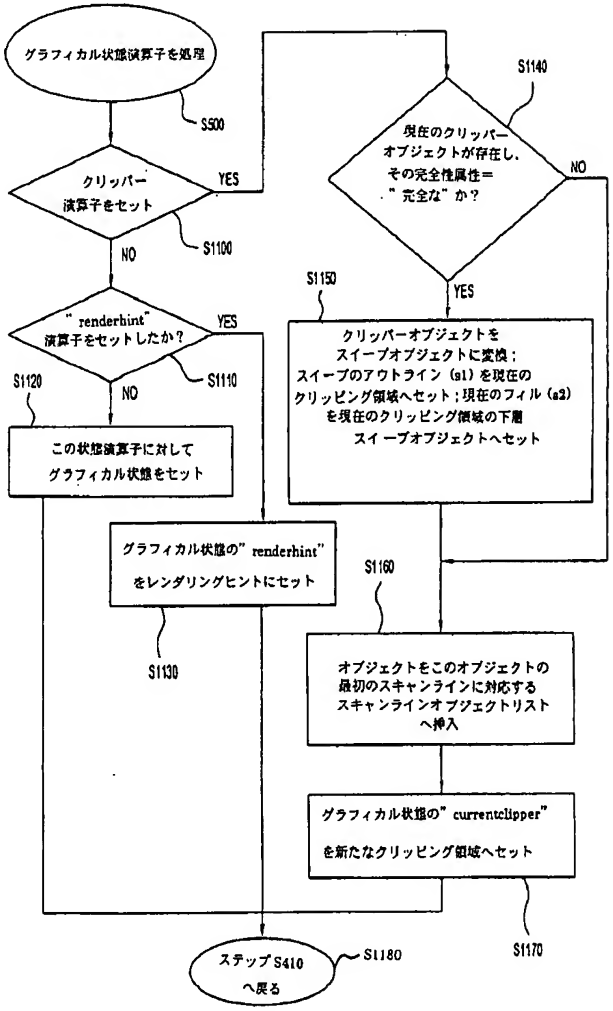
【図15】


カラー演算子
を処理
S460
S800
サンプリ
ングされた画像
があるか？
YES
NO
S860
画像データ及びIOT
解像度に基づいて画像
解像度除数を計算
S810
この一定
カラーは、レンダ
リングタグがセット
されていないパレット
エントリをすでに
有するか？
YES
NO
S870
グラフィカル状態演算子処理で
発生された" currenttranform"
及び画像除数に従って画像データ
を回転及びスケーリング
S830
S820
このカラーモデル及び画素
データを使用して新たな
パレットエントリを確立
同じカラーモデル及び
画素データを有する
パレットエントリを検出
このカラーモデル及びサンプリ
ング画像データを使用して新た
なパレットエントリを確立
S880
ハッシング関数をこの画素
データへ適用してパレット
インデックスを計算
S840
ハッシング関数を画像データ
S－サイズ（高さ）に適用
してパレットインデックス
を計算
S890
このパレットエントリを
インデックスで決定された
パレットスロットへ挿入
S850
このパレットエントリを
インデックスによって
決定されたスロット
でパレットへ挿入
S900
グラフィカル状態の
" currentcolor" をこの
パレットエントリへセット
S910
ステップS410へ戻る
S920

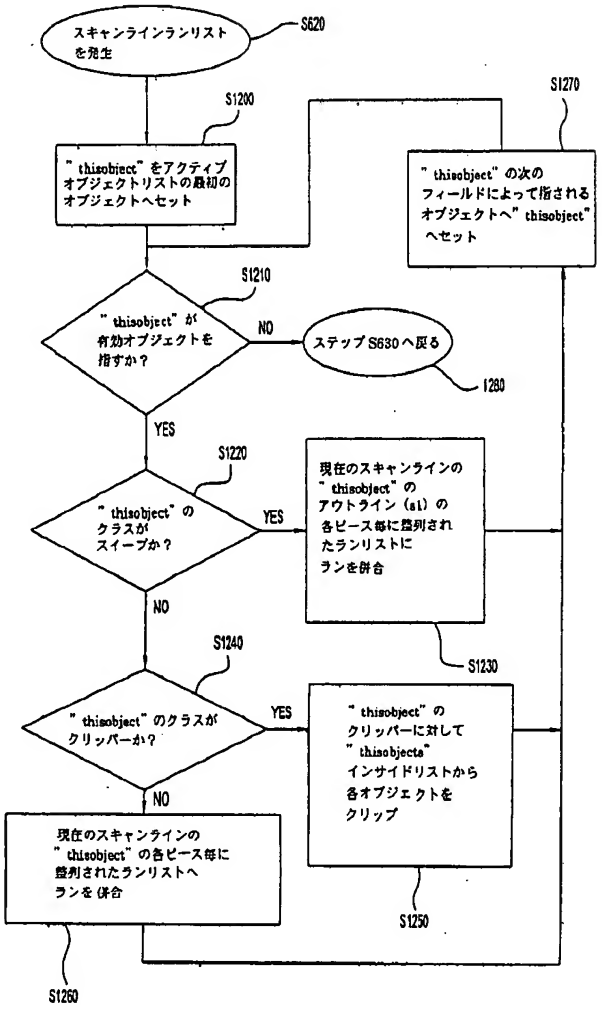
【図16】


マスキング
演算子を処理
S480
S1000
" renderhint" が
グラフィカル状態にセット
されたか？
YES
NO
S1010
マスキング演算子からオブジェクト
指定レンダリングタグを決定
S1020
" renderhint" からオブジェクト
指定レンダリングタグを決定
グラフィカル状態の" currentcolor"
のレンダリングタグを更新
S1030
プリミティブオブジェクトへのスキャン変換
S1040
S1080
S1050
プリミティブオブジェクト
がまだあるか？
NO
ステップS410へ戻る
YES
この演算子から次のプリミ
ティブオブジェクトを得る
S1060
プリミティブオブジェクトを処理
S1070

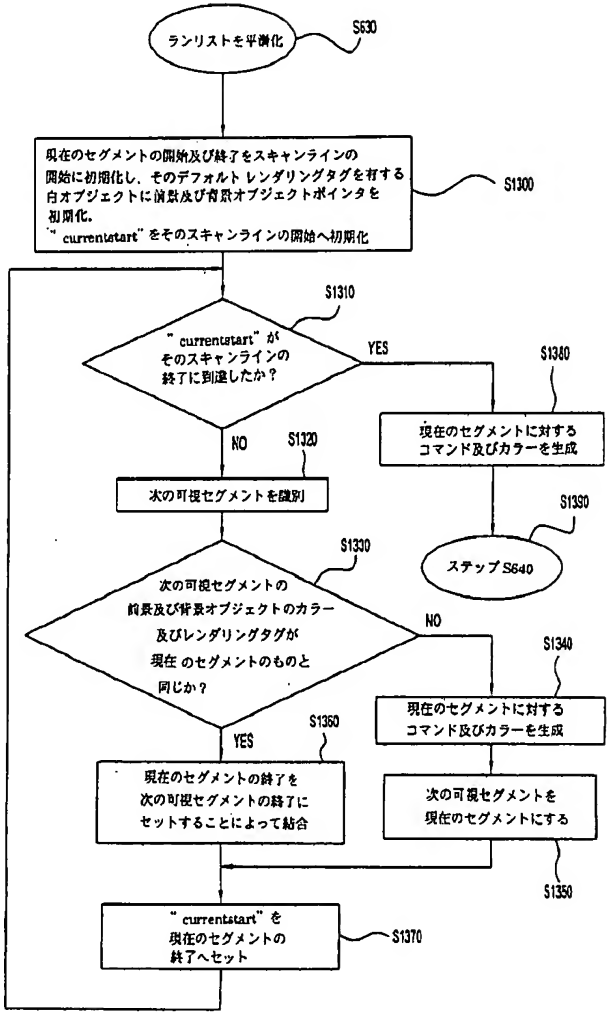
【図17】


グラフィカル状態演算子を処理
S500
クリッパー
演算子をセット
YES
S1100
NO
" renderhint"
演算子をセットしたか？
YES
S1110
NO
S1120
この状態演算子に対して
グラフィカル状態をセット
グラフィカル状態の" renderhint"
をレンダリングヒントにセット
S1130
S1140
現在のクリッパー
オブジェクトが存在し、
その完全性属性＝
" 完全な" か？
NO
YES
S1150
クリッパーオブジェクトを
スイープオブジェクトに変換；
スイープのアウトライン（s1）を現在の
クリッピング領域へセット；現在のフィル（s2）
を現在のクリッピング領域の下層
スイープオブジェクトへセット
S1160
オブジェクトをこのオブジェクトの
最初のスキャンラインに対応する
スキャンラインオブジェクトリスト
へ挿入
グラフィカル状態の" currentclipper"
を新たなクリッピング領域へセット
S1170
ステップS410
へ戻る
S1180

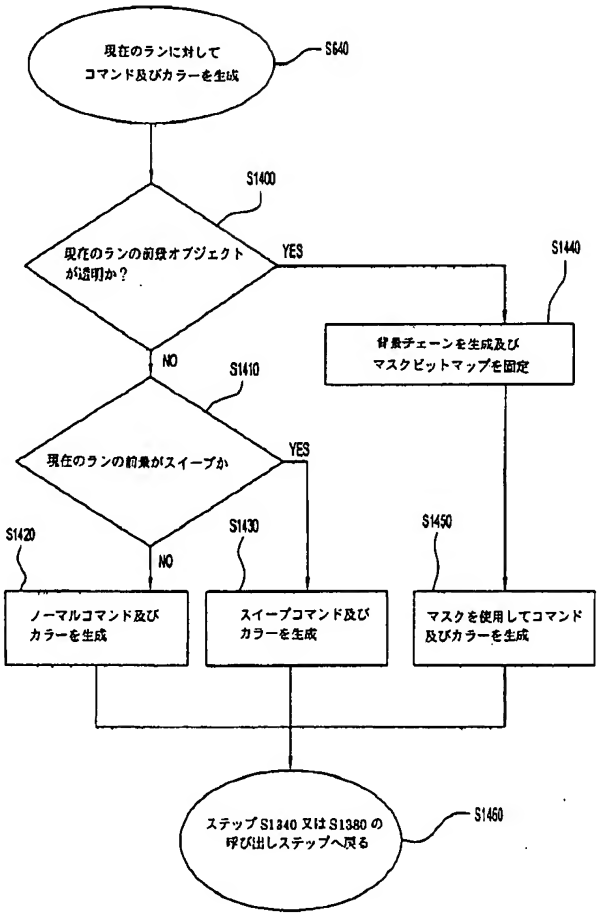
【図１８】


スキャンラインランリストを発生
S620
S1200
" thisobject" をアクティブオブジェクトリストの最初のオブジェクトへセット
S1270
" thisobject" の次のフィールドによって指されるオブジェクトへ" thisobject" へセット
S1210
" thisobject" が有効オブジェクトを指すか？
NO
ステップS630へ戻る
1280
YES
S1220
" thisobject" のクラスがスイープか？
YES
現在のスキャンラインの" thisobject" のアウトライン（s1）の各ピース毎に整列されたランリストにランを併合
S1230
NO
S1240
" thisobject" のクラスがクリッパーか？
YES
" thisobject" のクリッパーに対して" thisobjecta" インサイドリストから各オブジェクトをクリップ
S1250
NO
現在のスキャンラインの" thisobject" の各ピース毎に整列されたランリストへランを併合
S1260

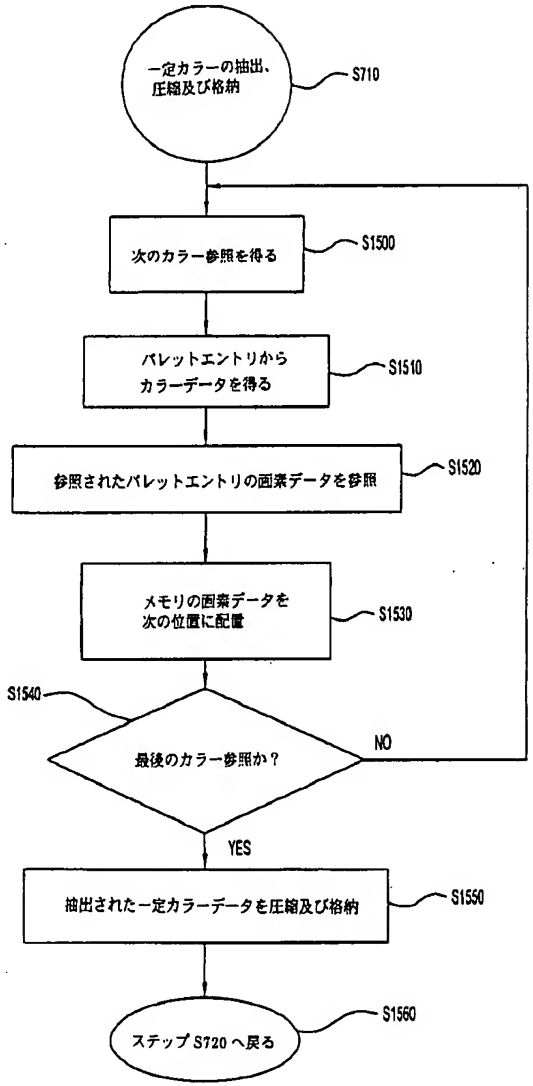
【図１９】


ランリストを平滑化
S630
現在のセグメントの開始及び終了をスキャンラインの開始に初期化し、そのデフォルトレンダリングタグを有する白オブジェクトに前景及び背景オブジェクトポインタを初期化。" currentstart" をそのスキャンラインの開始へ初期化
S1300
S1310
" currentstart" がそのスキャンラインの終了に到達したか？
YES
S1340
現在のセグメントに対するコマンド及びカラーを生成
NO
S1320
次の可視セグメントを識別
S1390
ステップS640
S1330
次の可視セグメントの前景及び背景オブジェクトのカラー及びレンダリングタグが現在のセグメントのものと同じか？
NO
S1340
現在のセグメントに対するコマンド及びカラーを生成
YES
S1360
現在のセグメントの終了を次の可視セグメントの終了にセットすることによって結合
次の可視セグメントを現在のセグメントにする
S1350
" currentstart" を現在のセグメントの終了へセット
S1370

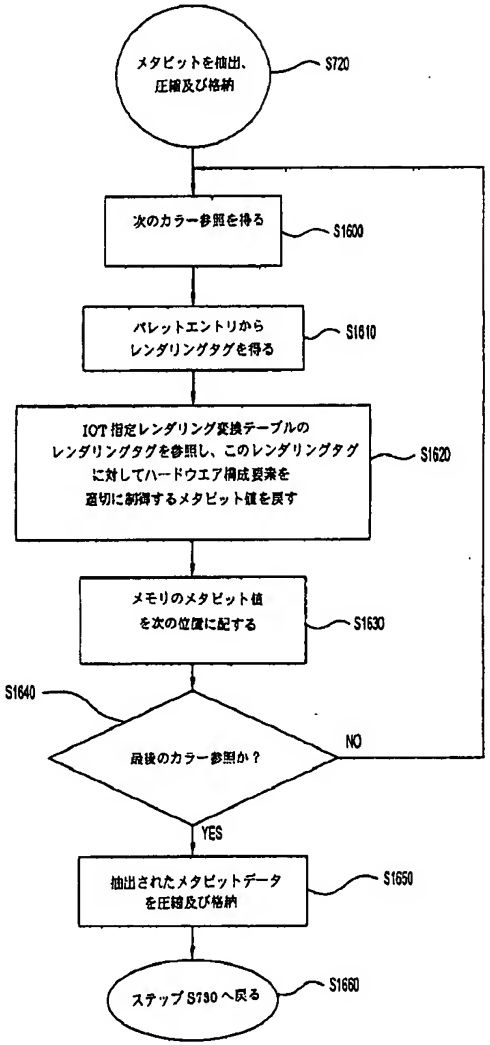
【図２０】


現在のランに対して
コマンド及びカラーを生成
S640
S1400
現在のランの前景オブジェクト
が透明か？
YES
S1440
背景チェーンを生成及び
マスクビットマップを固定
NO
S1410
現在のランの前景がスイープか
YES
S1420
NO
S1430
S1450
ノーマルコマンド及び
カラーを生成
スイープコマンド及び
カラーを生成
マスクを使用してコマンド
及びカラーを生成
ステップS1340又はS1380の
呼び出しステップへ戻る
S1460

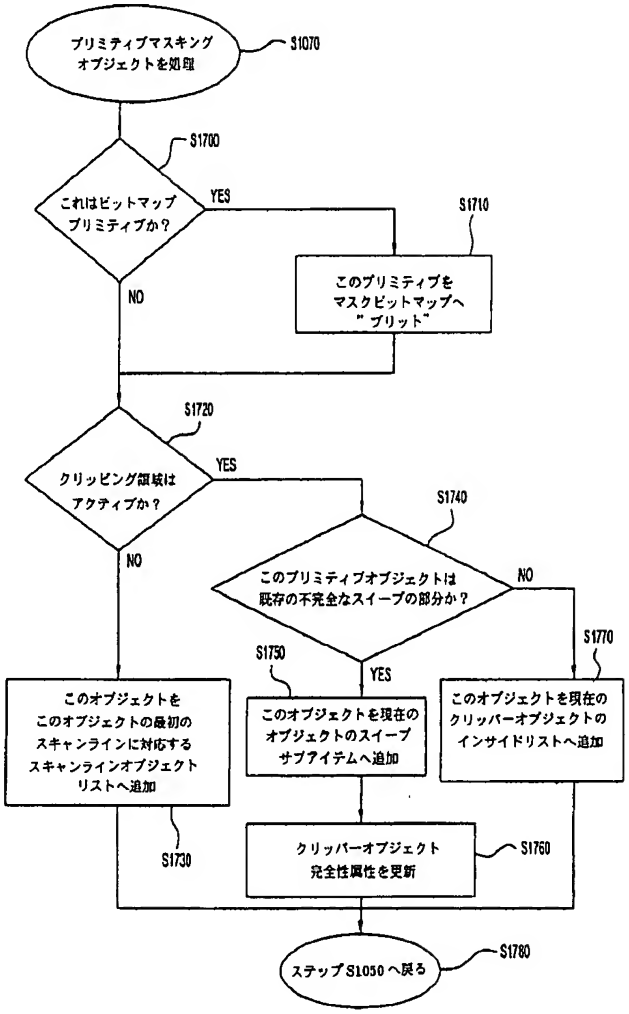
【図２１】


一定カラーの抽出、
圧縮及び格納
S710
次のカラー参照を得る
S1500
パレットエントリから
カラーデータを得る
S1510
参照されたパレットエントリの画素データを参照
S1520
メモリの画素データを
次の位置に配置
S1530
S1540
最後のカラー参照か？
NO
YES
抽出された一定カラーデータを圧縮及び格納
S1550
ステップS720へ戻る
S1560

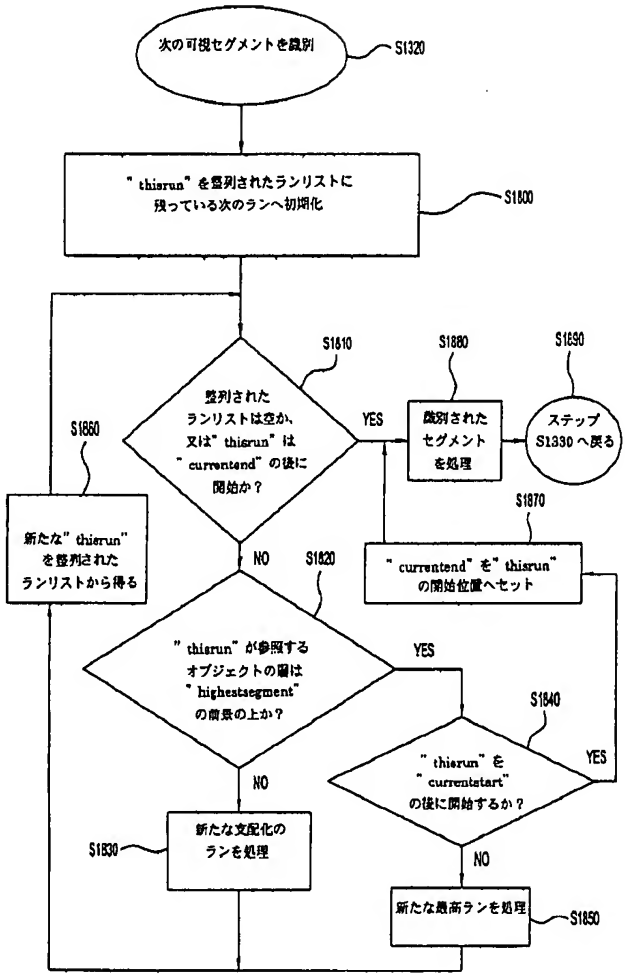
【図２２】


メタビットを抽出、
圧縮及び格納
S720
次のカラー参照を得る
S1600
パレットエントリから
レンダリングタグを得る
S1610
IOT指定レンダリング変換テーブルの
レンダリングタグを参照し、このレンダリングタグ
に対してハードウエア構成要素を
適切に制御するメタビット値を戻す
S1620
メモリのメタビット値
を次の位置に記する
S1630
S1640
最後のカラー参照か？
NO
YES
抽出されたメタビットデータ
を圧縮及び格納
S1650
ステップS730へ戻る
S1660

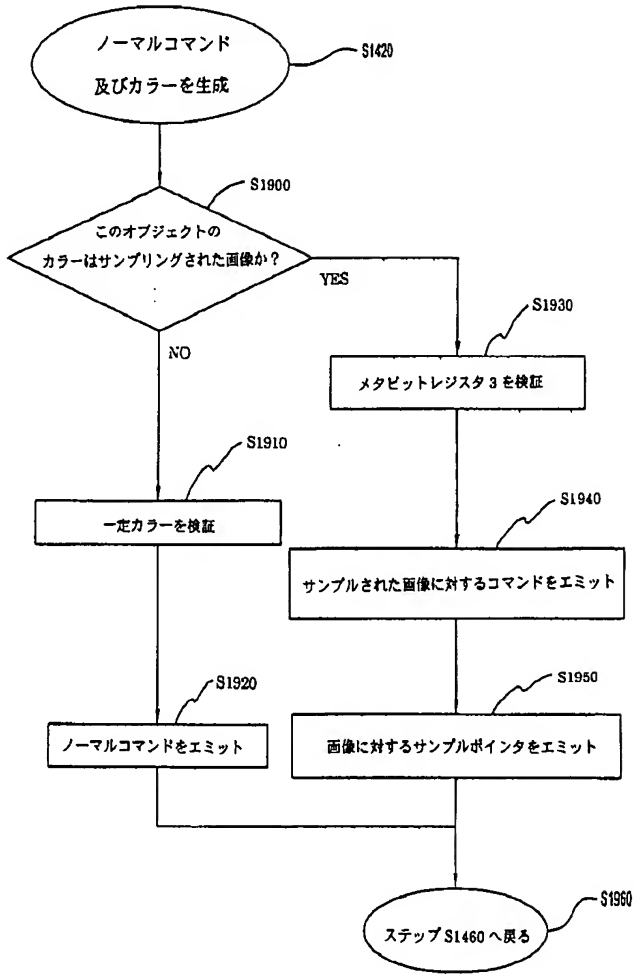
【図２３】


プリミティブマスキング
オブジェクトを処理
S1070
S1700
これはビットマップ
プリミティブか？
YES
NO
S1710
このプリミティブを
マスクビットマップへ
"ブリット"
S1720
クリッピング領域は
アクティブか？
YES
NO
S1740
このプリミティブオブジェクトは
既存の不完全なスイープの部分か？
NO
YES
S1730
このオブジェクトを
このオブジェクトの最初の
スキャンラインに対応する
スキャンラインオブジェクト
リストへ追加
S1730
このオブジェクトを現在の
オブジェクトのスイープ
サブアイテムへ追加
S1770
このオブジェクトを現在の
クリッパーオブジェクトの
インサイドリストへ追加
クリッパーオブジェクト
完全性属性を更新
S1760
ステップS1050へ戻る
S1780

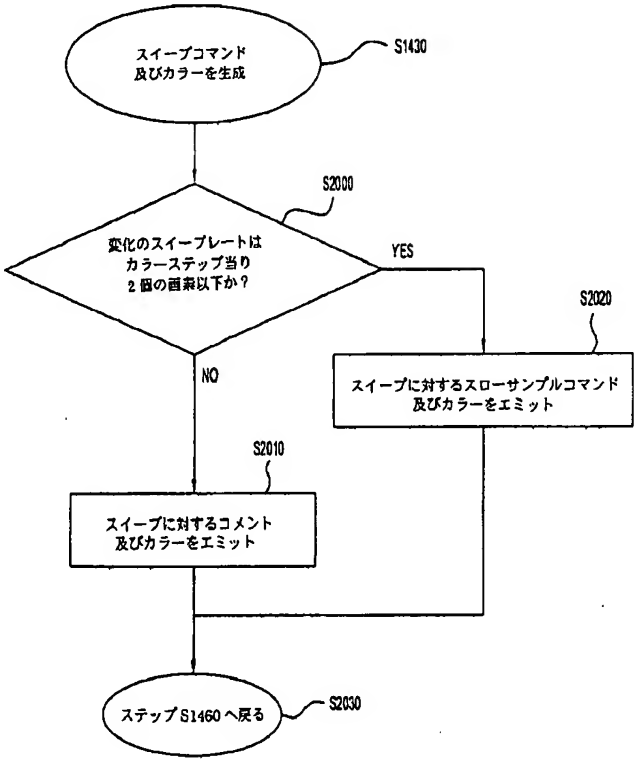
【図２４】


次の可視セグメントを識別
S1320
" thisrun" を整列されたランリストに残っている次のランへ初期化
S1800
整列されたランリストは空か、又は" thisrun" は " currentend" の後に開始か？
S1810
YES
識別されたセグメントを処理
S1880
ステップS1330へ戻る
S1890
NO
" thisrun" が参照するオブジェクトの層は " highestsegment" の前景の上か？
S1820
YES
NO
新たな支配化のランを処理
S1830
" thierun" を " currentstart" の後に開始するか？
S1840
YES
NO
新たな最高ランを処理
S1850
" currentend" を" thisrun" の開始位置へセット
S1870
新たな" thierun" を整列されたランリストから得る
S1860

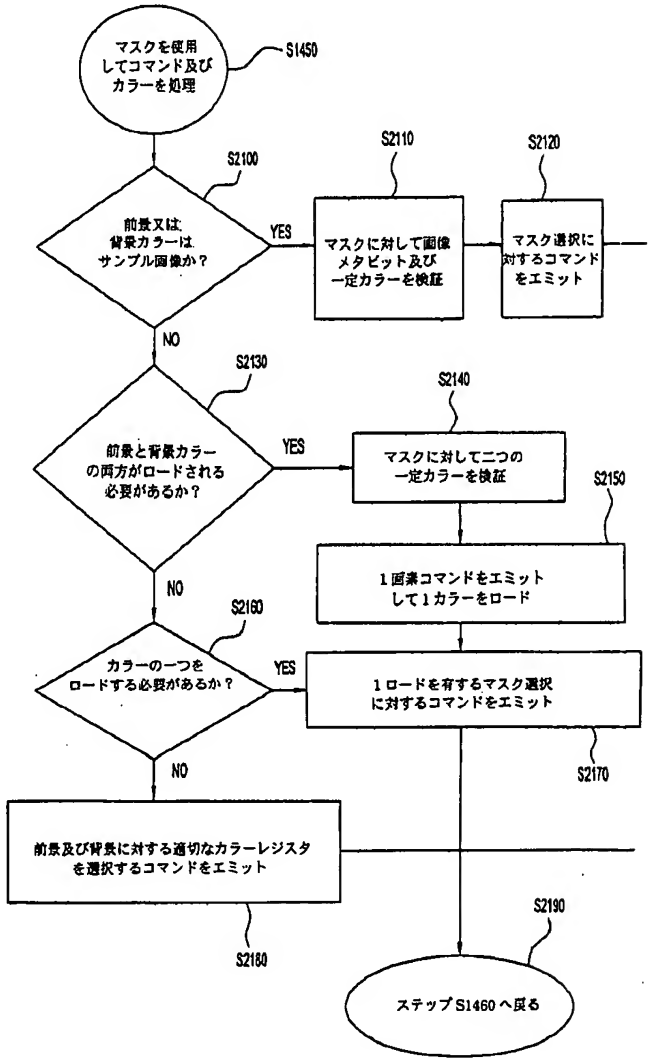
【図２５】


ノーマルコマンド及びカラーを生成
S1420
このオブジェクトのカラーはサンプリングされた画像か？
S1900
YES
NO
メタビットレジスタ３を検証
S1930
一定カラーを検証
S1910
サンプルされた画像に対するコマンドをエミット
S1940
ノーマルコマンドをエミット
S1920
画像に対するサンプルポインタをエミット
S1950
ステップS1460へ戻る
S1960

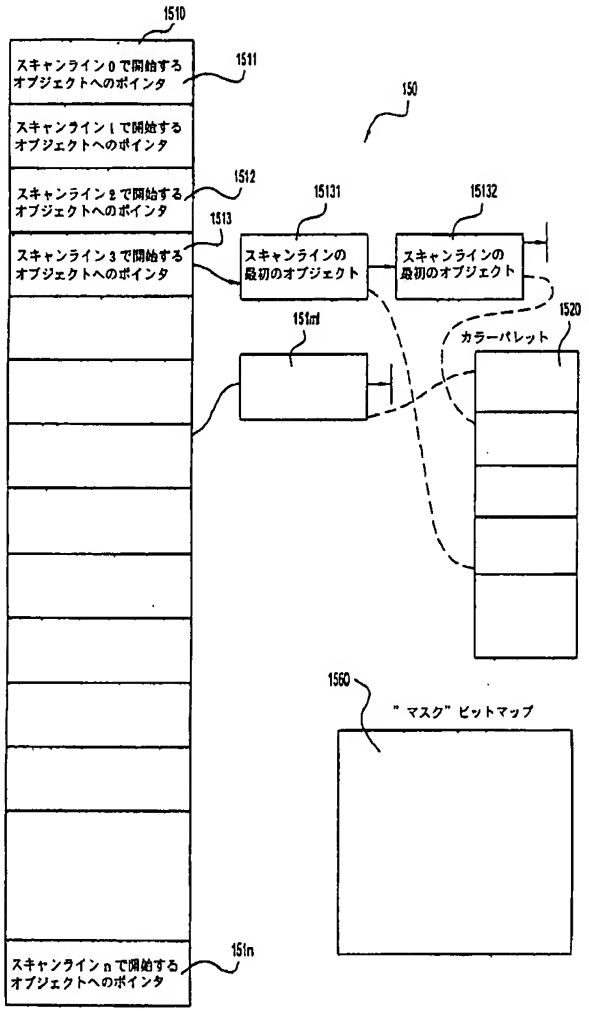
【図２６】


スイープコマンド
及びカラーを生成
S1430
S2000
変化のスイープレートは
カラーステップ当り
２個の画素以下か？
YES
NO
S2020
スイープに対するスローサンプルコマンド
及びカラーをエミット
S2010
スイープに対するコメント
及びカラーをエミット
ステップS1460へ戻る
S2030

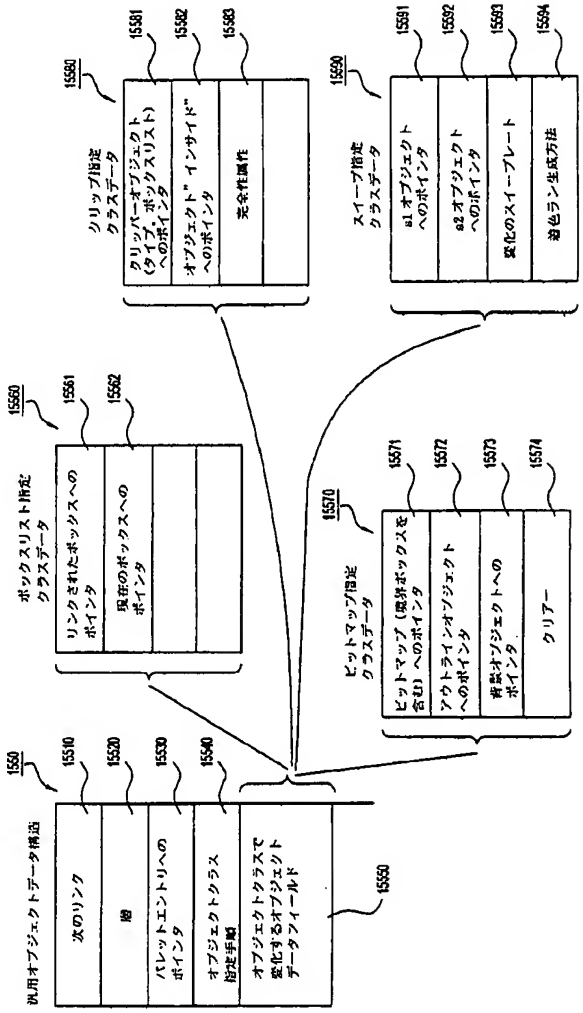
【図２７】


マスクを使用
してコマンド及び
カラーを処理
S1450
S2100
前景又は
背景カラーは
サンプル画像か？
YES
NO
S2110
マスクに対して画像
メタビット及び
一定カラーを検証
S2120
マスク選択に
対するコマンド
をエミット
S2130
前景と背景カラー
の両方がロードされる
必要があるか？
YES
NO
S2140
マスクに対して二つの
一定カラーを検証
S2150
１画素コマンドをエミット
して１カラーをロード
S2160
カラーの一つを
ロードする必要があるか？
YES
NO
１ロードを有するマスク選択
に対するコマンドをエミット
S2170
前景及び背景に対する適切なカラーレジスタ
を選択するコマンドをエミット
S2180
S2190
ステップS1460へ戻る

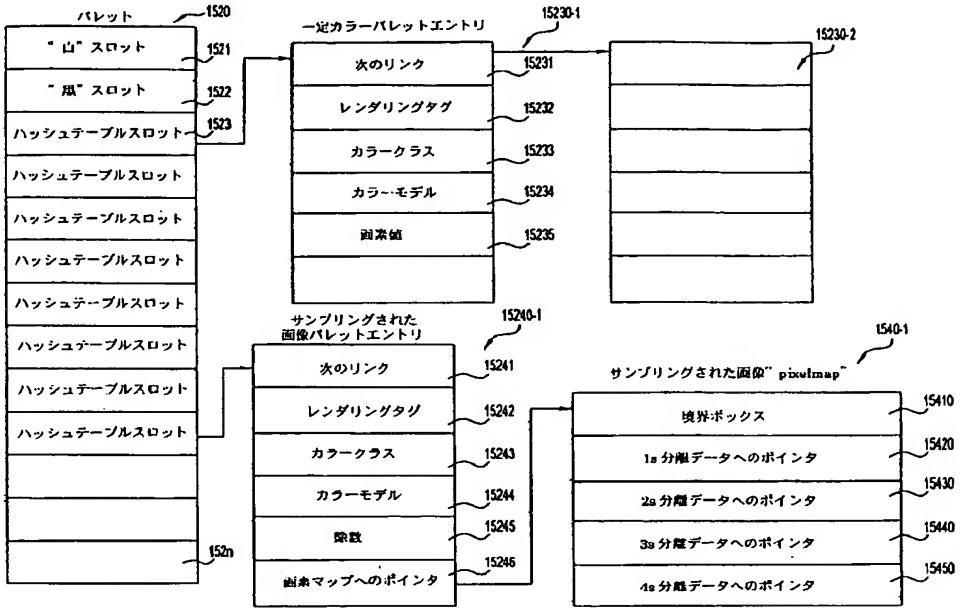
【図２８】


1510
スキャンライン０で開始するオブジェクトへのポインタ
1511
スキャンライン１で開始するオブジェクトへのポインタ
1512
スキャンライン２で開始するオブジェクトへのポインタ
1513
スキャンライン３で開始するオブジェクトへのポインタ
150
15131
スキャンラインの最初のオブジェクト
15132
スキャンラインの最初のオブジェクト
151m
1520
カラーパレット
1560
" マスク" ビットマップ
151n
スキャンラインｎで開始するオブジェクトへのポインタ

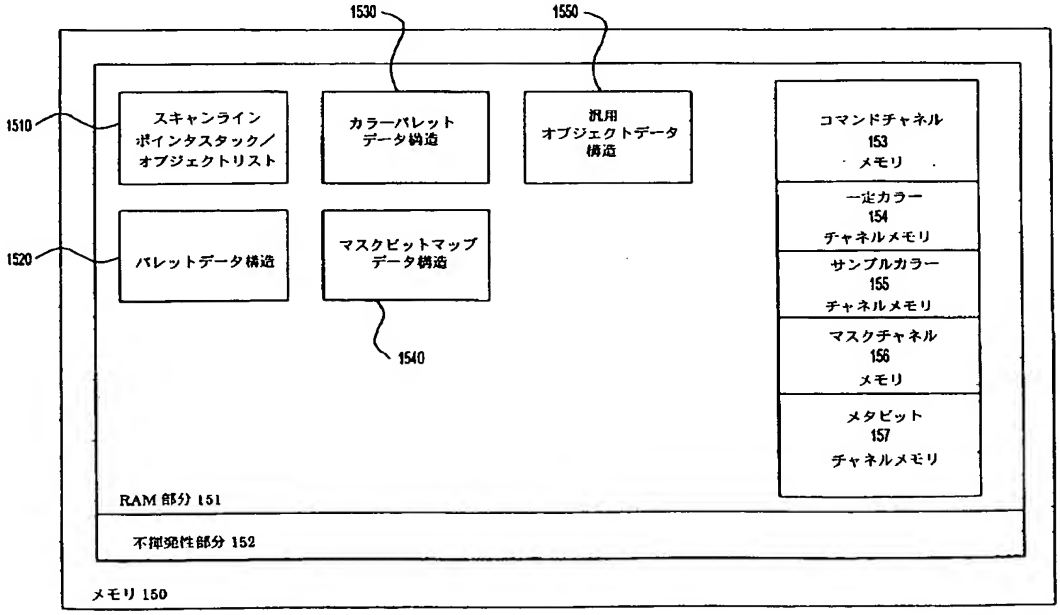
【図２９】


15581
15582
15583
15580
クリップ指定クラスデータ
クリッパーオブジェクト（クライアント・ボックスリスト）へのポインタ
オブジェクト" インサイド" へのポインタ
完全性属性
15591
15592
15593
15594
15590
スイープ指定クラスデータ
s1オブジェクトへのポインタ
s2オブジェクトへのポインタ
変化のスイープレート
塗色ラン生成方法
15560
15561
15562
ボックスリスト指定クラスデータ
リンクされたボックスへのポインタ
現在のボックスへのポインタ
15571
15572
15573
15574
15570
ビットマップ指定クラスデータ
ビットマップ（境界ボックスを含む）へのポインタ
アウトラインオブジェクトへのポインタ
背景オブジェクトへのポインタ
クリアー
1550
15510
15520
15530
15540
汎用オブジェクトデータ構造
次のリンク
線
パレットエントリへのポインタ
オブジェクトクラス指定手順
オブジェクトクラスで変化するオブジェクトデータフィールド
15550

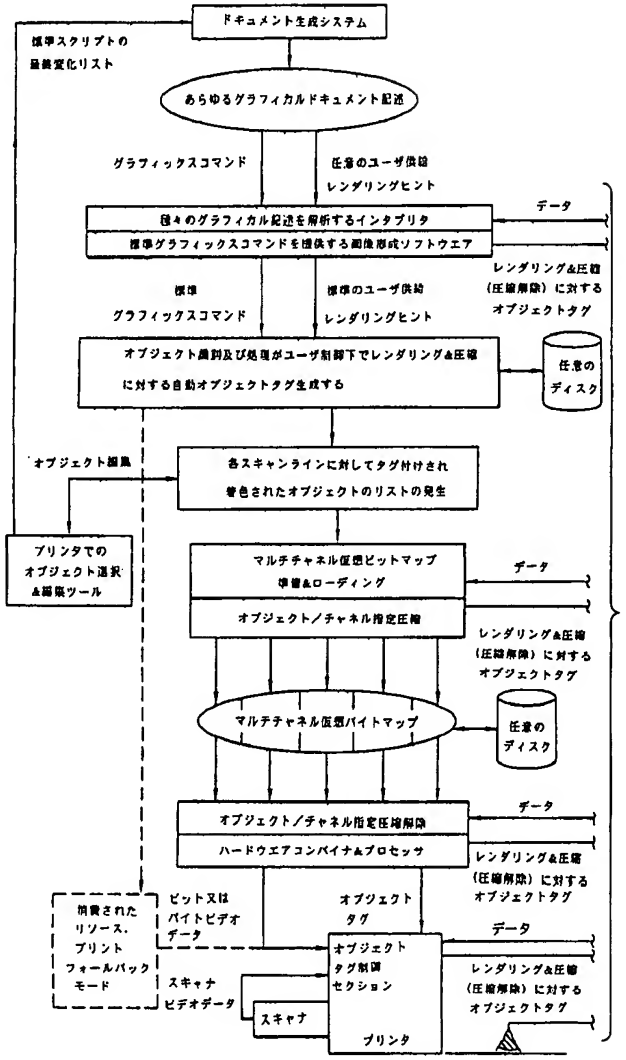
【図30】


パレット
1520
" 山" スロット
1521
" 風" スロット
1522
ハッシュテーブルスロット
1523
ハッシュテーブルスロット
ハッシュテーブルスロット
ハッシュテーブルスロット
ハッシュテーブルスロット
ハッシュテーブルスロット
ハッシュテーブルスロット
ハッシュテーブルスロット
152n
一定カラーパレットエントリ
15230-1
次のリンク
15231
レンダリングタグ
15232
カラークラス
15233
カラーモデル
15234
画素値
15235
15230-2
サンプリングされた
画像パレットエントリ
15240-1
次のリンク
15241
レンダリングタグ
15242
カラークラス
15243
カラーモデル
15244
階数
15245
画素マップへのポインタ
15246
サンプリングされた画像" pixelmap"
1540-1
境界ボックス
15410
1s 分離データへのポインタ
15420
2s 分離データへのポインタ
15430
3s 分離データへのポインタ
15440
4s 分離データへのポインタ
15450

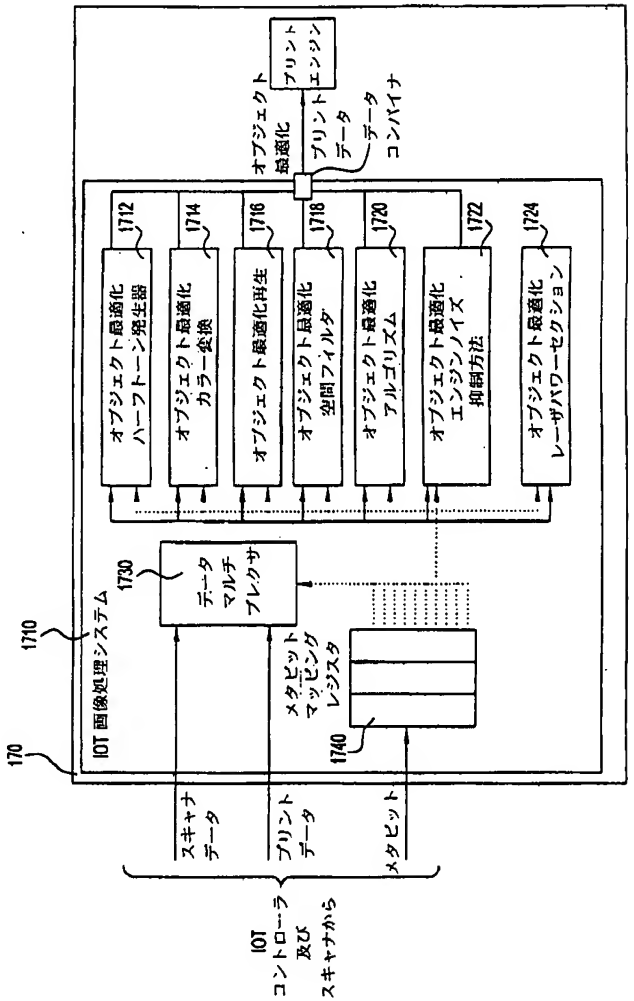
【図31】


1530
1550
1510
スキャンライン
ポインタスタック/
オブジェクトリスト
カラーパレット
データ構造
汎用
オブジェクトデータ
構造
コマンドチャネル
153
メモリ
1520
パレットデータ構造
マスクビットマップ
データ構造
1540
一定カラー
154
チャネルメモリ
サンプルカラー
155
チャネルメモリ
マスクチャネル
156
メモリ
メタビット
157
チャネルメモリ
RAM 部分 151
不揮発性部分 152
メモリ 150

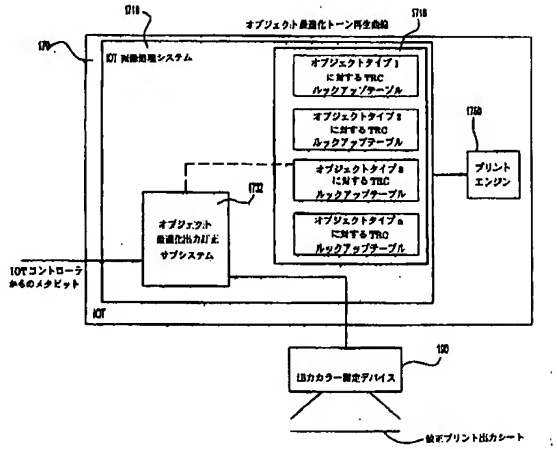
【図32】


ドキュメント生成システム
標準スクリプトの最適変化リスト
あらゆるグラフィカルドキュメント記述
グラフィックスコマンド
任意のユーザ供給レンダリングヒント
種々のグラフィカル記述を解析するインタプリタ
標準グラフィックスコマンドを提供する画像形成ソフトウエア
データ
標準グラフィックスコマンド
標準のユーザ供給レンダリングヒント
レンダリング&圧縮（圧縮解除）に対するオブジェクトタグ
オブジェクト識別及び処理がユーザ制御下でレンダリング&圧縮に対する自動オブジェクトタグ生成する
任意のディスク
オブジェクト識別
各スキャンラインに対してタグ付けされ着色されたオブジェクトのリストの発生
プリンタでのオブジェクト選択&編集ツール
マルチチャネル仮想ビットマップ準備&ローディング
データ
オブジェクト/チャネル指定圧縮
レンダリング&圧縮（圧縮解除）に対するオブジェクトタグ
マルチチャネル仮想バイトマップ
任意のディスク
オブジェクト/チャネル指定圧縮解除
ハードウエアコンパイナ&プロセッサ
データ
レンダリング&圧縮（圧縮解除）に対するオブジェクトタグ
消費されたリソース、プリントフォールバックモード
ビット又はバイトビデオデータ
オブジェクトタグ
オブジェクトタグ制御セクション
スキャナビデオデータ
スキャナ
プリンタ
データ
レンダリング&圧縮（圧縮解除）に対するオブジェクトタグ

【図34】


プリントエンジン
オブジェクト最適化
プリントデータ
データコンパイナ
1712
オブジェクト最適化ハーフトーン発生器
1714
オブジェクト最適化カラー変換
1716
オブジェクト最適化再生
1718
オブジェクト最適化空間フィルタ
1720
オブジェクト最適化アルゴリズム
1722
オブジェクト最適化エンジンノイズ抑制方法
1724
オブジェクト最適化レーザパワーセクション
1730
データマルチプレクサ
1710
IOT画像処理システム
メタビットマッピングレジスタ
1740
170
スキャナデータ
プリントデータ
メタビット
IOTコントローラ及びスキャナから

【図３５】


1718
オブジェクト最適化トーン再生曲線
1716
1710
IOT 画像処理システム
オブジェクトタイプ 1
に対する TRC
ルックアップテーブル
オブジェクトタイプ 2
に対する TRC
ルックアップテーブル
1720
プリント
エンジン
オブジェクトタイプ 3
に対する TRC
ルックアップテーブル
1722
オブジェクト
最適化出力訂正
サブシステム
IOTコントローラ
からのメタビット
IOT
120
出力カラー測定デバイス
被測プリント出力シート